GigaDevice Semiconductor Inc.

GD32E11x
ARM® Cortex®-M4 32-bit MCU

Firmware Library
User Guide

Revison 1.5

(Apr. 2026)

Table of Contents

Table of Contents 2
List of Figures 5
List of Tables 6
1. Introduction 20
1.1. Rules of User Manual and Firmware Library 20
1.1.1. Peripherals 20
1.1.2. Naming rules 21
2. Firmware Library Overview 22
2.1. File Structure of Firmware Library 22
2.1.1. Examples Folder 23
2.1.2. Firmware Folder 23
2.1.3. Template Folder 23
2.1.4. Utilities Folder 26
2.2. File descriptions of Firmware Library 26
3. Firmware Library of Standard Peripherals 28
3.1. Overview of Firmware Library of Standard Peripherals 28
3.2. ADC 28
3.2.1. Descriptions of Peripheral registers 28
3.2.2. Descriptions of Peripheral functions 29
3.3. BKP 57
3.3.1. Descriptions of Peripheral registers 57
3.3.2. Descriptions of Peripheral functions 57
3.4. CRC 68
3.4.1. Descriptions of Peripheral registers 68
3.4.2. Descriptions of Peripheral functions 69
3.5. CTC 73
3.5.1. Descriptions of Peripheral registers 73
3.5.2. Descriptions of Peripheral functions 73
3.6. DAC 86
3.6.1. Descriptions of Peripheral registers 86
3.6.2. Descriptions of Peripheral functions 87
3.7. DBG 101
3.7.1. Descriptions of Peripheral registers 101
3.7.2. Descriptions of Peripheral functions 101

3.8. DMA 106
3.8.1. Descriptions of Peripheral registers 106
3.8.2. Descriptions of Peripheral functions 107
3.9. EXMC 126
3.9.1. Descriptions of Peripheral registers 126
3.9.2. Descriptions of Peripheral functions 127
3.10. EXTI 132
3.10.1. Descriptions of Peripheral registers 132
3.10.2. Descriptions of Peripheral functions 132
3.11. FMC 140
3.11.1. Descriptions of Peripheral registers 140
3.11.2. Descriptions of Peripheral functions 140
3.12. FWDGT 160
3.12.1. Descriptions of Peripheral registers 160
3.12.2. Descriptions of Peripheral functions 160
3.13. GPIO 165
3.13.1. Descriptions of Peripheral registers 165
3.13.2. Descriptions of Peripheral functions 166
3.14. I2C 179
3.14.1. Descriptions of Peripheral registers 179
3.14.2. Descriptions of Peripheral functions 180
3.15. MISC 205
3.15.1. Descriptions of Peripheral registers 205
3.15.2. Descriptions of Peripheral functions 206
3.16. PMU 212
3.16.1. Descriptions of Peripheral registers 212
3.16.2. Descriptions of Peripheral functions 213
3.17. RCU 220
3.17.1. Descriptions of Peripheral registers 220
3.17.2. Descriptions of Peripheral functions 221
3.18. RTC 252
3.18.1. Descriptions of Peripheral registers 252
3.18.2. Descriptions of Peripheral functions 252
3.19. SPI 261
3.19.1. Descriptions of Peripheral registers 262
3.19.2. Descriptions of Peripheral functions 262
3.20. TIMER 287
3.20.1. Descriptions of Peripheral registers 287
3.20.2. Descriptions of Peripheral functions 288

3.21. USART 343
3.21.1. Descriptions of Peripheral registers 344
3.21.2. Descriptions of Peripheral functions 344
3.22. WWDGT 379
3.22.1. Descriptions of Peripheral registers 379
3.22.2. Descriptions of Peripheral functions 379
4. Revision history 384

List of Figures

Figure 2-1. File structure of firmware library of GD32E11x 22
Figure 2-2. Select peripheral example files 24
Figure 2-3. Copy the peripheral example files 25
Figure 2-4. Open the project file 25
Figure 2-5. Compile-debug-download 26

List of Tables

Table 1-1. Peripherals 20
Table 2-1. Function descriptions of Firmware Library 26
Table 3-1. Peripheral function format of Firmware Library 28
Table 3-2. ADC Registers 28
Table 3-3. ADC firmware function 29
Table 3-4. Function adc_deinit 30
Table 3-5. Function adc_mode_config 31
Table 3-6. Function adc_special_function_config 32
Table 3-7. Function adc_data_alignment_config 32
Table 3-8. Function adc_enable 33
Table 3-9. Function adc_disable 34
Table 3-10. Function adc_calibration_enable 34
Table 3-11. Function adc_tempsensor_vrefint_enable 35
Table 3-12. Function adc_tempsensor_vrefint_disable 35
Table 3-13. Function adc_resolution_config 36
Table 3-14. Function adc_oversample_mode_config 36
Table 3-15. Function adc_oversample_mode_enable 38
Table 3-16. Function adc_oversample_mode_disable 38
Table 3-17. Function adc_dma_mode_enable 39
Table 3-18. Function adc_dma_mode_disable 39
Table 3-19. Function adc_discontinuous_mode_config 40
Table 3-20. Function adc_channel_length_config 41
Table 3-21. Function adc_regular_channel_config 42
Table 3-22. Function adc_inserted_channel_config 43
Table 3-23. Function adc_inserted_channel_offset_config 44
Table 3-24. Function adc_external_trigger_source_config 44
Table 3-25. Function adc_external_trigger_config 46
Table 3-26. Function adc_software_trigger_enable 47
Table 3-27. Function adc_regular_data_read 47
Table 3-28. Function adc_inserted_data_read 48
Table 3-29. Function adc_sync_mode_convert_value_read 49
Table 3-30. Function adc_watchdog_single_channel_enable 49
Table 3-31. Function adc_watchdog_group_channel_enable 50
Table 3-32. Function adc_watchdog_disable 50
Table 3-33. Function adc_watchdog_threshold_config 51
Table 3-34. Function adc_flag_get 52
Table 3-35. Function adc_flag_clear 52
Table 3-36. Function adc_regular_software_startconv_flag_get 53
Table 3-37. Function adc_inserted_software_startconv_flag_get 54
Table 3-38. Function adc_interrupt_flag_get 54

Table 3-39. Function adc_interrupt_flag_clear 55
Table 3-40. Function adc_interrupt _enable 56
Table 3-41. Function adc_interrupt_disable 56
Table 3-42. BKP Registers 57
Table 3-43. BKP firmware function 57
Table 3-44. Function bkp_deinit 58
Table 3-45. Function bkp_data_write 58
Table 3-46. Function bkp_data_read 59
Table 3-47. Function bkp_rtc_calibration_output_enable 60
Table 3-48. Function bkp_rtc_calibration_output_disable 60
Table 3-49. Function bkp_rtc_signal_output_enable 61
Table 3-50. Function bkp_rtc_signal_output_disable 61
Table 3-51. Function bkp_rtc_output_select 62
Table 3-52. Function bkp_rtc_clock_output_select 62
Table 3-53. Function bkp_rtc_clock_calibration_direction_select 63
Table 3-54. Function bkp_rtc_calibration_value_set 63
Table 3-55. Function bkp_tamper_detection_enable 64
Table 3-56. Function bkp_tamper_detection_disable 64
Table 3-57. Function bkp_tamper_active_level_set 65
Table 3-58. Function bkp_interrupt_enable 65
Table 3-59. Function bkp_interrupt_disable 66
Table 3-60. Function bkp_flag_get 66
Table 3-61. Function bkp_flag_clear 67
Table 3-62. Function bkp_interrupt_flag_get 67
Table 3-63. Function bkp_interrupt_flag_clear 68
Table 3-64. CRC Registers 69
Table 3-65. CRC firmware function 69
Table 3-66. Function crc_deinit 69
Table 3-67. Function crc_data_register_reset 70
Table 3-68. Function crc_data_register_read 70
Table 3-69. Function crc_free_data_register_read 71
Table 3-70. Function crc_free_data_register_write 71
Table 3-71. Function crc_single_data_calculate 72
Table 3-72. Function crc_block_data_calculate 72
Table 3-73. CTC Registers 73
Table 3-74. CTC firmware function 73
Table 3-75. Function ctc_deinit 74
Table 3-76. Function ctc_counter_enable 75
Table 3-77. Function ctc_counter_disable 75
Table 3-78. Function ctc_irc48m_trim_value_config 76
Table 3-79. Function ctc_software_refsource_pulse_generate 76
Table 3-80. Function ctc_hardware_trim_mode_config 77
Table 3-81. Function ctc_refsource_polarity_config 77
Table 3-82. Function ctc_refsource_signal_select 78

Table 3-83. Function ctc_refsource_prescaler_config 78
Table 3-84. Function ctc_clock_limit_value_config 79
Table 3-85. Function ctc_counter_reload_value_config 80
Table 3-86. Function ctc_counter_capture_value_read 80
Table 3-87. Function ctc_counter_direction_read 81
Table 3-88. Function ctc_counter_reload_value_read 81
Table 3-89. Function ctc_irc48m_trim_value_read 82
Table 3-90. Function ctc_interrupt_enable 82
Table 3-91. Function ctc_interrupt_disable 83
Table 3-92. Function ctc_interrupt_flag_get 83
Table 3-93. Function ctc_interrupt_flag_clear 84
Table 3-94. Function ctc_flag_get 85
Table 3-95. Function ctc_flag_clear 86
Table 3-96. DAC Registers 86
Table 3-97. DAC firmware function 87
Table 3-98. Function dac_deinit 87
Table 3-99. Function dac_enable 88
Table 3-100. Function dac_disable 89
Table 3-101. Function dac_dma_enable 89
Table 3-102. Function dac_dma_disable 90
Table 3-103. Function dac_output_buffer_enable 90
Table 3-104. Function dac_output_buffer_disable 91
Table 3-105. Function dac_output_value_get 92
Table 3-106. Function dac_data_set 92
Table 3-107. Function dac_trigger_enable 93
Table 3-108. Function dac_trigger_disable 93
Table 3-109. Function dac_trigger_source_config 94
Table 3-110. Function dac_software_trigger_enable 95
Table 3-111. Function dac_wave_mode_config 96
Table 3-112. Function dac_lfsr_noise_config 96
Table 3-113. Function dac_triangle_noise_config 97
Table 3-114. Function dac_concurrent_enable 98
Table 3-115. Function dac_concurrent_disable 98
Table 3-116. Function dac_concurrent_software_trigger_enable 99
Table 3-117. Function dac_concurrent_output_buffer_enable 99
Table 3-118. Function dac_concurrent_output_buffer_disable 100
Table 3-119. Function dac_concurrent_data_set 100
Table 3-120. DBG Registers 101
Table 3-121. DBG firmware function 101
Table 3-122. Enum dbg_periph_enum 102
Table 3-123. Function dbg_id_get 102
Table 3-124. Function dbg_low_power_enable 103
Table 3-125. Function dbg_low_power_disable 103
Table 3-126. Function dbg_periph_enable 104

Table 3-127. Function dbg_periph_disable 105
Table 3-128. Function dbg_trace_pin_enable 105
Table 3-129. Function dbg_trace_pin_disable 106
Table 3-130. DMA Registers 106
Table 3-131. DMA firmware function 107
Table 3-132. Structure dma_parameter_struct 108
Table 3-133. Function dma_deinit 108
Table 3-134. Function dma_struct_para_init 109
Table 3-135. Function dma_init 109
Table 3-136. Function dma_circulation_enable 110
Table 3-137. Function dma_circulation_disable 111
Table 3-138. Function dma_memory_to_memory_enable 111
Table 3-139. Function dma_memory_to_memory_disable 112
Table 3-140. Function dma_channel_enable 112
Table 3-141. Function dma_channel_disable 113
Table 3-142. Function dma_periph_address_config 114
Table 3-143. Function dma_memory_address_config 114
Table 3-144. Function dma_transfer_number_config 115
Table 3-145. Function dma_transfer_number_get 116
Table 3-146. Function dma_priority_config 116
Table 3-147. Function dma_memory_width_config 117
Table 3-148. Function dma_periph_width_config 118
Table 3-149. Function dma_memory_increase_enable 119
Table 3-150. Function dma_memory_increase_disable 119
Table 3-151. Function dma_periph_increase_enable 120
Table 3-152. Function dma_periph_increase_disable 120
Table 3-153. Function dma_transfer_direction_config 121
Table 3-154. Function dma_flag_get 122
Table 3-155. Function dma_flag_clear 123
Table 3-156. Function dma_interrupt_flag_get 123
Table 3-157. Function dma_interrupt_flag_clear 124
Table 3-158. Function dma_interrupt_enable 125
Table 3-159. Function dma_interrupt_disable 126
Table 3-160. EXMC Registers 127
Table 3-161. EXMC firmware function 127
Table 3-162. Structure exmc_norsram_timing_parameter_struct 127
Table 3-163. Structure exmc_norsram_parameter_struct 127
Table 3-164. Function exmc_norsram_deinit 128
Table 3-165. Function exmc_norsram_init 128
Table 3-166. Function exmc_norsram_struct_para_init 130
Table 3-167. Function exmc_norsram_enable 130
Table 3-168. Function exmc_norsram_disable 131
Table 3-169. Function exmc_norsram_page_size_config 131
Table 3-170. EXTI Registers 132

Table 3-171. EXTI firmware function 132
Table 3-172. Enum exti_line_enum 133
Table 3-173. Enum exti_mode_enum 133
Table 3-174. Enum exti_trig_type_enum 134
Table 3-175. Function exti_deinit 134
Table 3-176. Function exti_init 134
Table 3-177. Function exti_interrupt_enable 135
Table 3-178. Function exti_interrupt_disable 135
Table 3-179. Function exti_event_enable 136
Table 3-180. Function exti_event_disable 136
Table 3-181. Function exti_software_interrupt_enable 137
Table 3-182. Function exti_software_interrupt_disable 137
Table 3-183. Function exti_flag_get 138
Table 3-184. Function exti_flag_clear 138
Table 3-185. Function exti_interrupt_flag_get 139
Table 3-186. Function exti_interrupt_flag_clear 139
Table 3-187. FMC Registers 140
Table 3-188. FMC firmware function 140
Table 3-189. Enum fmc_state_enum 141
Table 3-190. Function fmc_wscnt_set 142
Table 3-191. Function fmc_prefetch_enable 142
Table 3-192. Function fmc_prefetch_disable 143
Table 3-193. Function fmc_ibus_enable 143
Table 3-194. Function fmc_ibus_disable 144
Table 3-195. Function fmc_dbus_enable 144
Table 3-196. Function fmc_dbus_disable 145
Table 3-197. Function fmc_ibus_reset 145
Table 3-198. Function fmc_dbus_reset 146
Table 3-199. Function fmc_program_width_set 146
Table 3-200. Function fmc_unlock 147
Table 3-201. Function fmc_lock 147
Table 3-202. Function fmc_page_erase 148
Table 3-203. Function fmc_mass_erase 148
Table 3-204. Function fmc_doubleword_program 149
Table 3-205. Function fmc_word_program 149
Table 3-206. Function ob_unlock 150
Table 3-207. Function ob_lock 150
Table 3-208. Function ob_erase 151
Table 3-209. Function ob_write_protection_enable 151
Table 3-210. Function ob_security_protection_config 152
Table 3-211. Function ob_user_write 152
Table 3-212. Function ob_data_program 153
Table 3-213. Function ob_user_get 153
Table 3-214. Function ob_data_program 154

Table 3-215. Function ob_write_protection_get 154
Table 3-216. Function ob_security_protection_flag_get 155
Table 3-217. Function fmc_interrupt_enable 155
Table 3-218. Function fmc_interrupt_disable 156
Table 3-219. Function fmc_flag_get 156
Table 3-220. Function fmc_flag_clear 157
Table 3-221. Function fmc_interrupt_flag_get 158
Table 3-222. Function fmc_interrupt_flag_clear 158
Table 3-223. Function fmc_state_get 159
Table 3-224. Function fmc_ready_wait 159
Table 3-225. FWDGT Registers 160
Table 3-226. FWDGT firmware function 160
Table 3-227. Function fwdgt_write_ensable 161
Table 3-228. Function fwdgt_write_disable 161
Table 3-229. Function fwdgt_enable 162
Table 3-230. Function fwdgt_prescaler_value_config 162
Table 3-231. Function fwdgt_reload_value_config 163
Table 3-232. Function fwdgt_counter_reload 163
Table 3-233. Function fwdgt_config 164
Table 3-234. Function fwdgt_flag_get fwdgt_write_disable 164
Table 3-235. GPIO Registers 165
Table 3-236. GPIO firmware function 166
Table 3-237. Function gpio_deinit 166
Table 3-238. Function gpio_afio_deinit 167
Table 3-239. Function gpio_init 167
Table 3-240. Function gpio_bit_set 169
Table 3-241. Function gpio_bit_reset 169
Table 3-242. Function gpio_bit_write 170
Table 3-243. Function gpio_port_write 171
Table 3-244. Function gpio_input_bit_get 171
Table 3-245. Function gpio_input_port_get 172
Table 3-246. Function gpio_output_bit_get 172
Table 3-247. Function gpio_output_port_get 173
Table 3-248. Function gpio_pin_remap_config 173
Table 3-249. Function gpio_exti_source_select 175
Table 3-250. Function gpio_event_output_config 176
Table 3-251. Function gpio_event_output_enable 176
Table 3-252. Function gpio_event_output_disable 177
Table 3-253. Function gpio_pin_lock 177
Table 3-254. Function gpio_compensation_config 178
Table 3-255. Function gpio_compensation_flag_get 178
Table 3-256. I2C Registers 179
Table 3-257. I2C firmware function 180
Table 3-258. Enum i2c_flag_enum 181

Table 3-259. Enum i2c_interrupt_flag_enum 181
Table 3-260. Enum i2c_interrupt_enum 182
Table 3-261. Function i2c_deinit 182
Table 3-262. Function i2c_clock_config 183
Table 3-263. Function i2c_mode_addr_config 183
Table 3-264. Function i2c_smbus_type_config 184
Table 3-265. Function i2c_ack_config 185
Table 3-266. Function i2c_ackpos_config 186
Table 3-267. Function i2c_master_addressing 186
Table 3-268. Function i2c_dualaddr_enable 187
Table 3-269. Function i2c_dualaddr_disable 187
Table 3-270. Function i2c_enable 188
Table 3-271. Function i2c_disable 188
Table 3-272. Function i2c_start_on_bus 189
Table 3-273. Function i2c_stop_on_bus 189
Table 3-274. Function i2c_data_transmit 190
Table 3-275. Function i2c_data_receive 191
Table 3-276. Function i2c_dma_config 191
Table 3-277. Function i2c_dma_last_transfer_config 192
Table 3-278. Function i2c_stretch_scl_low_config 192
Table 3-279. Function i2c_slave_response_to_gcall_config 193
Table 3-280. Function i2c_software_reset_config 194
Table 3-281. Function i2c_pec_config 194
Table 3-282. Function i2c_pec_transfer_config 195
Table 3-283. Function i2c_pec_value_get 195
Table 3-284. Function i2c_smbus_alert_config 196
Table 3-285. Function i2c_smbus_arp_config 197
Table 3-286. Function i2c_sam_enable 197
Table 3-287. Function i2c_sam_disable 198
Table 3-288. Function i2c_sam_timeout_enable 198
Table 3-289. Function i2c_sam_timeout_disable 199
Table 3-290. Function i2c_flag_get 199
Table 3-291. Function i2c_flag_clear 201
Table 3-292. Function i2c_interrupt_enable 201
Table 3-293. Function i2c_interrupt_disable 202
Table 3-294. Function i2c_interrupt_flag_get 203
Table 3-295. Function i2c_interrupt_flag_clear 204
Table 3-296. NVIC Registers 205
Table 3-297. SysTick Registers 206
Table 3-298. IRQn_Type 206
Table 3-299. MISC firmware function 208
Table 3-300. Function nvic_priority_group_set 208
Table 3-301. Function nvic_irq_enable 209
Table 3-302. Function nvic_irq_disable 209

Table 3-303. Function nvic_vector_table_set 210
Table 3-304. Function system_lowpower_set 210
Table 3-305. Function system_lowpower_reset 211
Table 3-306. Function systick_clksource_set 212
Table 3-307. PMU Registers 212
Table 3-308. PMU firmware function 213
Table 3-309. Function pmu_deinit 213
Table 3-310. Function pmu_lvd_select 214
Table 3-311. Function pmu_ldo_output_select 214
Table 3-312. Function pmu_lvd_disable 215
Table 3-313. Function pmu_to_sleepmode 215
Table 3-314. Function pmu_to_deepsleepmode 216
Table 3-315. Function pmu_to_standbymode 216
Table 3-316. Function pmu_wakeup_pin_enable 217
Table 3-317. Function pmu_wakeup_pin_disable 217
Table 3-318. Function pmu_backup_write_enable 218
Table 3-319. Function pmu_backup_write_disable 218
Table 3-320. Function pmu_flag_get 219
Table 3-321. Function pmu_flag_clear 219
Table 3-322. RCU Registers 220
Table 3-323. RCU firmware function 221
Table 3-324. rcu_periph_enum 222
Table 3-325. rcu_periph_sleep_enum 223
Table 3-326. rcu_periph_reset_enum 223
Table 3-327. rcu_flag_enum 224
Table 3-328. rcu_int_flag_enum 225
Table 3-329. rcu_int_flag_clear_enum 226
Table 3-330. rcu_int_enum 226
Table 3-331. rcu_osci_type_enum 226
Table 3-332. rcu_clock_freq_enum 227
Table 3-333. Function rcu_deinit 227
Table 3-334. Function rcu_periph_clock_enable 227
Table 3-335. Function rcu_periph_clock_disable 228
Table 3-336. Function rcu_periph_clock_sleep_enable 228
Table 3-337. Function rcu_periph_clock_sleep_disable 229
Table 3-338. Function rcu_periph_reset_enable 229
Table 3-339. Function rcu_periph_reset_disable 230
Table 3-340. Function rcu_bkp_reset_enable 230
Table 3-341. Function rcu_bkp_reset_disable 231
Table 3-342. Function rcu_system_clock_source_config 231
Table 3-343. Function rcu_system_clock_source_get 232
Table 3-344. Function rcu_ahb_clock_config 232
Table 3-345. Function rcu_apb1_clock_config 233
Table 3-346. Function rcu_apb2_clock_config 234

Table 3-347. Function rcu_ckout0_config 234
Table 3-348. Function rcu_pll_config 235
Table 3-349. Function rcu_pllpresel_config 236
Table 3-350. Function rcu_predv0_config 237
Table 3-351. Function rcu_predv1_config 237
Table 3-352. Function rcu_pll1_config 238
Table 3-353. Function rcu_pll2_config 238
Table 3-354. Function rcu_adc_clock_config 239
Table 3-355. Function rcu_usb_clock_config 240
Table 3-356. Function rcu_rtc_clock_config 241
Table 3-357. Function rcu_i2s1_clock_config 241
Table 3-358. Function rcu_i2s2_clock_config 242
Table 3-359. Function rcu_ck48m_clock_config 242
Table 3-360. Function rcu_flag_get 243
Table 3-361. Function rcu_all_reset_flag_clear 243
Table 3-362. Function rcu_interrupt_flag_get 244
Table 3-363. Function rcu_interrupt_flag_clear 244
Table 3-364. Function rcu_interrupt_enable 245
Table 3-365. Function rcu_interrupt_disable 245
Table 3-366. Function rcu_lxtal_drive_capability_config 246
Table 3-367. Function rcu_osci_stab_wait 247
Table 3-368. Function rcu_osci_on 247
Table 3-369. Function rcu_osci_off 248
Table 3-370. Function rcu_osci_bypass_mode_enable 248
Table 3-371. Function rcu_osci_bypass_mode_disable 249
Table 3-372. Function rcu_hxtal_clock_monitor_enable 249
Table 3-373. Function rcu_hxtal_clock_monitor_disable 250
Table 3-374. Function rcu_irc8m_adjust_value_set 250
Table 3-375. Function rcu_deepsleep_voltage_set 251
Table 3-376. Function rcu_clock_freq_get 251
Table 3-377. RTC Registers 252
Table 3-378. RTC firmware function 252
Table 3-379. Function rtc_configuration_mode_enter 253
Table 3-380. Function rtc_configuration_mode_exit 253
Table 3-381. Function rtc_counter_set 254
Table 3-382. Function rtc_prescaler_set 254
Table 3-383. Function rtc_lwoff_wait 255
Table 3-384. Function rtc_register_sync_wait 256
Table 3-385. Function rtc_alarm_config 256
Table 3-386. Function rtc_counter_get 257
Table 3-387. Function rtc_divider_get 257
Table 3-388. Function rtc_flag_get 258
Table 3-389. Function rtc_flag_clear 258
Table 3-390. Function rtc_ interrupt_flag_get 259

Table 3-391. Function rtc_interrupt_flag_clear 260
Table 3-392. Function rtc_interrupt_enable 260
Table 3-393. Function rtc_interrupt_disable 261
Table 3-394. SPI/I2S Registers 262
Table 3-395. SPI/I2S firmware function 263
Table 3-396. spi_parameter_struct 264
Table 3-397. Function spi_i2s_deinit 264
Table 3-398. Function spi_struct_para_init 265
Table 3-399. Function spi_init 265
Table 3-400. Function spi_enable 266
Table 3-401. Function spi_disable 266
Table 3-402. Function i2s_init 267
Table 3-403. Function i2s_psc_config 268
Table 3-404. Function i2s_enable 269
Table 3-405. Function i2s_disable 270
Table 3-406. Function spi_nss_output_enable 270
Table 3-407. Function spi_nss_output_disable 271
Table 3-408. Function spi_nss_internal_high 271
Table 3-409. Function spi_nss_internal_low 272
Table 3-410. Function spi_dma_enable 272
Table 3-411. Function spi_dma_disable 273
Table 3-412. Function spi_i2s_data_frame_format_config 274
Table 3-413. Function spi_i2s_data_transmit 274
Table 3-414. Function spi_i2s_data_receive 275
Table 3-415. Function spi_bidirectional_transfer_config 275
Table 3-416. Function spi_crc_polynomial_set 276
Table 3-417. Function spi_crc_polynomial_get 277
Table 3-418. Function spi_crc_on 277
Table 3-419. Function spi_crc_off 278
Table 3-420. Function spi_crc_next 278
Table 3-421. Function spi_crc_get 279
Table 3-422. Function spi_ti_mode_enable 279
Table 3-423. Function spi_ti_mode_disable 280
Table 3-424. Function spi_nssp_mode_enable 280
Table 3-425. Function spi_nssp_mode_disable 281
Table 3-426. Function spi_quad_enable 281
Table 3-427. Function spi_quad_disable 282
Table 3-428. Function spi_quad_write_enable 282
Table 3-429. Function spi_quad_read_enable 283
Table 3-432. Function spi_i2s_interrupt_enable 283
Table 3-433. Function spi_i2s_interrupt_disable 284
Table 3-434. Function spi_i2s_interrupt_flag_get 285
Table 3-435. Function spi_i2s_flag_get 286
Table 3-436. Function spi_crc_error_clear 286

Table 3-437. TIMERx Registers 287
Table 3-438. TIMERx firmware function 288
Table 3-439. Structure timer_parameter_struct 290
Table 3-440. Structure timer_break_parameter_struct 290
Table 3-441. Structure timer_oc_parameter_struct 291
Table 3-442. Structure timer_ic_parameter_struct 291
Table 3-443. Function timer_deinit 292
Table 3-444. Function timer_struct_para_init 292
Table 3-445. Function timer_init 293
Table 3-446. Function timer_enable 293
Table 3-447. Function timer_disable 294
Table 3-448. Function timer_auto_reload_shadow_enable 294
Table 3-449. Function timer_auto_reload_shadow_disable 295
Table 3-450. Function timer_update_event_enable 295
Table 3-451. Function timer_update_event_disable 296
Table 3-452. Function timer_counter_alignment 297
Table 3-453. Function timer_counter_up_direction 297
Table 3-454. timer_counter_down_direction 298
Table 3-455. Function timer_prescaler_config 298
Table 3-456. Function timer_repetition_value_config 299
Table 3-457. Function timer_autoreload_value_config 300
Table 3-458. Function timer_counter_value_config 300
Table 3-459. Function timer_counter_read 301
Table 3-460. Function timer_prescaler_read 301
Table 3-461. Function timer_single_pulse_mode_config 302
Table 3-462. Function timer_update_source_config 303
Table 3-463. Function timer_dma_enable 303
Table 3-464. Function timer_dma_disable 304
Table 3-465. Function timer_channel_dma_request_source_select 305
Table 3-466. Function timer_dma_transfer_config 305
Table 3-467. Function timer_event_software_generate 307
Table 3-468. Function timer_break_struct_para_init 308
Table 3-469. Function timer_break_config 309
Table 3-470. Function timer_break_enable 309
Table 3-471. Function timer_break_disable 310
Table 3-472. Function timer_automatic_output_enable 311
Table 3-473. Function timer_automatic_output_disable 311
Table 3-474. Function timer_primary_output_config 312
Table 3-475. Function timer_channel_control_shadow_config 312
Table 3-476. Function timer_channel_control_shadow_update_config 313
Table 3-477. Function timer_channel_output_struct_para_init 314
Table 3-478. Function timer_channel_output_config 314
Table 3-479. Function timer_channel_output_mode_config 315
Table 3-480. Function timer_channel_output_pulse_value_config 316

Table 3-481. Function timer_channel_output_shadow_config 317
Table 3-482. Function timer_channel_output_fast_config 318
Table 3-483. Function timer_channel_output_clear_config 319
Table 3-484. Function timer_channel_output_polarity_config 319
Table 3-485. Function timer_channel_complementary_output_polarity_config 320
Table 3-486. Function timer_channel_output_state_config 321
Table 3-487. Function timer_channel_complementary_output_state_config 322
Table 3-488. Function timer_channel_input_struct_para_init 323
Table 3-489. Function timer_input_capture_config 323
Table 3-490. Function timer_channel_input_capture_prescaler_config 324
Table 3-491. Function timer_channel_capture_value_register_read 325
Table 3-492. Function timer_input_pwm_capture_config 326
Table 3-493. Function timer_hall_mode_config 327
Table 3-494. Function timer_input_trigger_source_select 327
Table 3-495. Function timer_master_output_trigger_source_select 328
Table 3-496. Function timer_slave_mode_select 329
Table 3-497. Function timer_master_slave_mode_config 330
Table 3-498. Function timer_external_trigger_config 331
Table 3-499. Function timer_quadrature_decoder_mode_config 332
Table 3-500. Function timer_internal_clock_config 333
Table 3-501. Function timer_internal_trigger_as_external_clock_config 333
Table 3-502. Function timer_external_trigger_as_external_clock_config 334
Table 3-503. Function timer_external_clock_mode0_config 335
Table 3-504. Function timer_external_clock_mode1_config 336
Table 3-505. Function timer_external_clock_mode1_disable 337
Table 3-506. Function timer_write_chxval_register_config 337
Table 3-507. Function timer_output_value_selection_config 338
Table 3-508. Function timer_interrupt_enable 339
Table 3-509. Function timer_interrupt_disable 339
Table 3-510. Function timer_interrupt_flag_get 340
Table 3-511. Function timer_interrupt_flag_clear 341
Table 3-512. Function timer_flag_get 342
Table 3-513. Function timer_flag_clear 343
Table 3-514. USART Registers 344
Table 3-515. USART firmware function 344
Table 3-516. usart_flag_enum 346
Table 3-517. usart_interrupt_flag_enum 346
Table 3-518. usart_interrupt_enum 347
Table 3-519. usart_invert_enum 347
Table 3-520. Function usart_deinit 347
Table 3-521. Function usart_baudrate_set 348
Table 3-522. Function usart_parity_config 348
Table 3-523. Function usart_word_length_set 349
Table 3-524. Function usart_stop_bit_set 350

Table 3-525. Function usart_enable 350
Table 3-526. Function usart_disable 351
Table 3-527. Function usart_transmit_config 351
Table 3-528. Function usart_receive_config 352
Table 3-529. Function usart_data_first_config 353
Table 3-530. Function usart_invert_config 353
Table 3-531. Function usart_receiver_timeout_enable 354
Table 3-532. Function usart_receiver_timeout_disable 355
Table 3-533. Function usart_receiver_timeout_threshold_config 355
Table 3-534. Function usart_data_transmit 356
Table 3-535. Function usart_data_receive 356
Table 3-536. Function usart_address_config 357
Table 3-537. Function usart_mute_mode_enable 358
Table 3-538. Function usart_mute_mode_disable 358
Table 3-539. Function usart_mute_mode_wakeup_config 359
Table 3-540. Function usart_lin_mode_enable 359
Table 3-541. Function usart_lin_mode_disable 360
Table 3-542. Function usart_lin_break_dection_length_config 360
Table 3-543. Function usart_send_break 361
Table 3-544. Function usart_halfduplex_enable 362
Table 3-545. Function usart_halfduplex_disable 362
Table 3-546. Function usart_synchronous_clock_enable 363
Table 3-547. Function usart_synchronous_clock_disable 363
Table 3-548. Function usart_synchronous_clock_config 364
Table 3-549. Function usart_guard_time_config 364
Table 3-550. Function usart_smartcard_mode_enable 365
Table 3-551. Function usart_smartcard_mode_disable 366
Table 3-552. Function usart_smartcard_mode_nack_enable 366
Table 3-553. Function usart_smartcard_mode_nack_disable 367
Table 3-554. Function usart_smartcard_autoretry_config 367
Table 3-555. Function usart_block_length_config 368
Table 3-556. Function usart_irda_mode_enable 368
Table 3-557. Function usart_irda_mode_disable 369
Table 3-558. Function usart_prescaler_config 369
Table 3-559. Function usart_irda_lowpower_config 370
Table 3-560. Function usart_hardware_flow_rts_config 370
Table 3-561. Function usart_hardware_flow_cts_config 371
Table 3-562. Function usart_dma_receive_config 372
Table 3-563. Function usart_dma_transmit_config 372
Table 3-564. Function usart_hardware_flow_coherence_config 373
Table 3-565. Function usart_flag_get 374
Table 3-566. Function usart_flag_clear 374
Table 3-567. Function usart_interrupt_enable 375
Table 3-568. Function usart_interrupt_disable 376

Table 3-569. Function usart_interrupt_flag_get 377
Table 3-570. Function usart_interrupt_flag_clear 378
Table 3-571. WWDGT Registers 379
Table 3-572. WWDGT firmware function 379
Table 3-573. Function wwdgt_deinit 380
Table 3-574. Function wwdgt_enable 380
Table 3-575. Function wwdgt_counter_update 381
Table 3-576. Function wwdgt_config 381
Table 3-577. Function wwdgt_interrupt_enable 382
Table 3-578. Function wwdgt_flag_get 382
Table 3-579. Function wwdgt_flag_clear 383
Table 4-1. Revision history 384

1. Introduction

This manual introduces firmware library of GD32E11x devices which are 32-bit microcontrollers based on the ARM processor.

The firmware library is a firmware function package, including program, data structure and macro definitions, all the performance features of peripherals of GD32E11x devices are involved in the package. The peripheral driving code and firmware examples on evaluation board are also included in firmware library. Users need not learn each peripherals in details and it's easy to apply a peripheral by using the firmware library. Using firmware library can greatly reduce programming time, thereby reducing development costs.

The driving code of each peripheral is concluded by a group of functions, which describes all the performance features of the peripheral. Users can drive a peripheral by a group of APIs (application programming interface), all the APIs are standardized about the code structure, function name and parameter names.

All the driving source code accord with MISRA-C:2004 standard (example files accord with extended ANSI-C standard), and will not be influenced by differences of IDEs, except the startup files which are written differently according to the IDEs.

The commonly used firmware library includes all the functions of all the peripherals, so the code size and the execution speed may not be the optimal. For most applications, users can use the library functions directly, while for the applications which are strict with the code size and execution speed, the firmware library can be used as the reference resource of how to configure a peripheral, and users adjust the code according to actual needs.

The overall structure of the firmware library user manual is shown as below:

- Rules of user manual and firmware library;
- Firmware library overview;
- Functions and registers descriptions of firmware library.

1.1. Rules of User Manual and Firmware Library

1.1.1. Peripherals

Table 1-1. Peripherals

Peripherals	Descriptions
ADC	Analog-to-digital converter
BKP	Backup registers
CRC	CRC calculation unit
CTC	Clock trim controller
DAC	Digital-to-analog converter
DBG	Debug

Peripherals	Descriptions
DMA	Direct memory access controller
EXMC	External memory controller
EXTI	Interrupt/event controller
FMC	Flash memory controller
FWDGT	Free watchdog timer
GPIO/AFIO	General-purpose and alternate-function I/Os
I2C	Inter-integrated circuit interface
MISC	Nested Vectored Interrupt Controller
PMU	Power management unit
RCU	Reset and clock unit
RTC	Real-time Clock
SPI/I2S	Serial peripheral interface/Inter-IC sound
TIMER	TIMER
USART	Universal synchronous/asynchronous receiver /transmitter
WWDGT	Window watchdog timer
USBFS	Universal serial bus full-speed interface

1.1.2. Naming rules

The firmware library naming rules are shown as below:

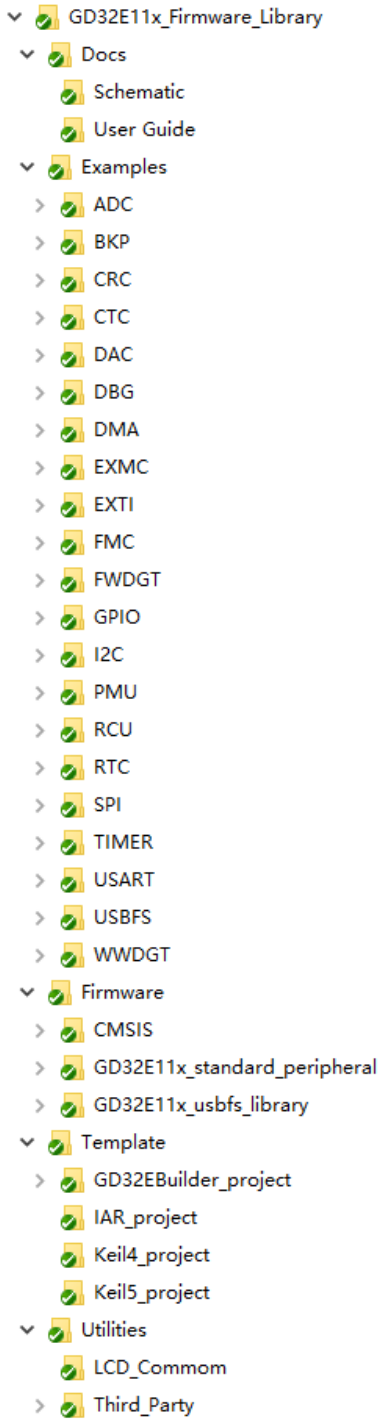
- The peripherals are shortened in XXX format, such as: ADC. More shorten information of peripherals refer to ***Peripherals***;
- The name of sourcefile and header file are started with "gd32e11x_", such as: gd32e11x_adc.h;
- The constants used only in one file should be defined in the used file; the constants used in many files should be defined in corresponding header file. All the constants are written in uppercase of English letters;
- Registers are handled as constants. The naming of them are written in uppercase of English letters. In most cases, register names are shortened accord with the user manual;
- Variables are written in lowercase, when concluded by several words, underlines should be adapted among words;
- The naming of peripheral functions are started with the peripheral abbreviation added with an underline, when the function name is concluded by several words, underlines should be adapted among words, and all the peripheral functions are written in lowercase.

2. Firmware Library Overview

2.1. File Structure of Firmware Library

GD32E11x_Firmware_Library, the file structure is shown as below:

Figure 2-1. File structure of firmware library of GD32E11x



2.1.1. Examples Folder

Examples folder, each of GD32 peripheral has a subfolder. Each subfolder contains one or more examples of the peripheral, to show how to use the peripheral correctly. Each of the example subfolder includes the files shown as below:

- readme.txt: the description and using guide of the example;
- gd32e11x_libopt.h: the header file configures all the peripherals used in the example, included by different "DEFINE" sentences (all the peripherals are enabled by default);
- gd32e11x_it.c: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- gd32e11x.it.h: the header file include all the prototypes of the interrupt service routines;
- systick.c: the source file include the precise time delay functions by using systick;
- systick.h: the header file include the prototype of the precise time delay functions by using systick;
- main.c: example code. Note: all the examples are not influenced by software IDEs.

2.1.2. Firmware Folder

Firmware folder includes all the subfolder and files which are the core part of the firmware:

- CMSIS subfolder includes the Cortex M4 kernel support files, the startup file based on the Cortex M4 kernel processor, the global header file of GD32E11x and system configuration file;
- GD32E11x_standard_peripheral subfolder:
  - Include subfolder includes all the header files of firmware libray, users need not modify this folder;
  - Source subfolder includes all the source files of firmware library, users need not modify this folder;
- GD32E11x_usbfs_library subfolder includes all the related files about USBFS peripheral:users need not modify this folder;

Note: All the codes accord with MISRA-C:2004 standard, and will not be influenced by different software IDEs.

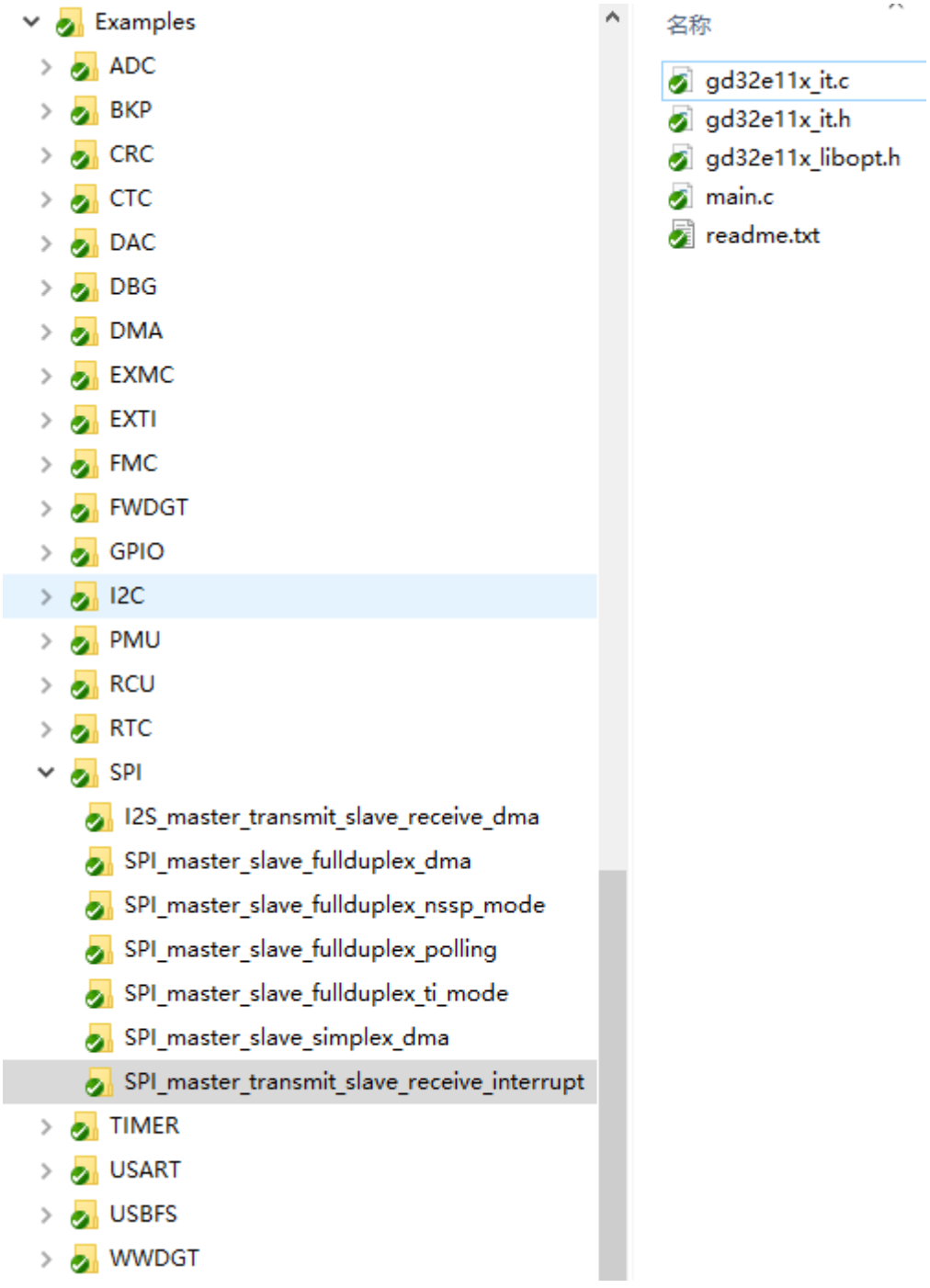
2.1.3. Template Folder

Template folder includes a simple demo of how to use LED, how to print by USART and use key to control, (IAR_project is run in IAR, Keil4_project is run in Keil4, Keil5_project is run in Keil5 and GD32EBuilder_project is run in GD32EmbeddedBuilder). User can use the project template to compile the formware examples, the steps are shown as below:

Select files

Open "Examples" folder, select the module to be tested, such as SPI, open "SPI" folder, select an example of SPI, such as "SPI_master_transmit_slave_receive_interrupt", shown as below:

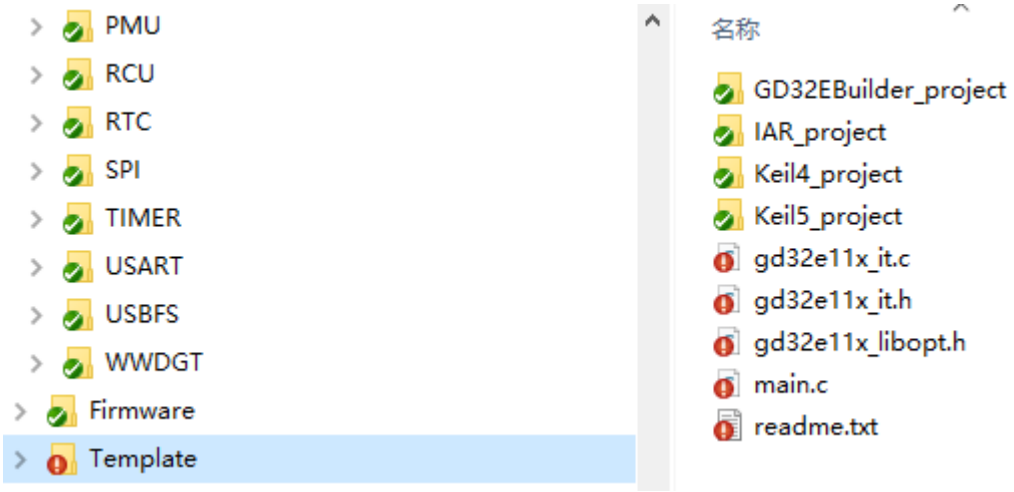
Figure 2-2. Select peripheral example files



Copy files

Open "Template" folder, keep the folders of " IAR_project", " Keil4_project" , " Keil5_project" and "GD32EBuilder_project", and delete the other files, then copy all the files in "SPI_master_transmit_slave_receive_interrupt" folder to the "Template" subfolder, shown as below:

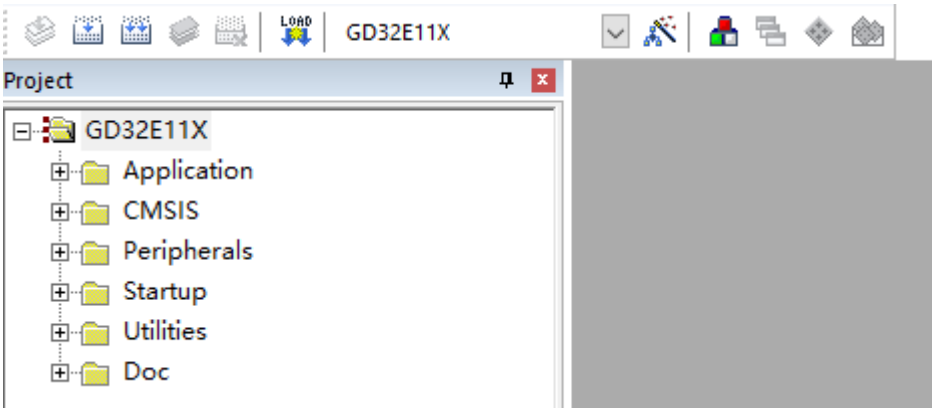
Figure 2-3. Copy the peripheral example files



Open a project

GD provides project in Keil4, Keil5, IAR, and GD32EBuilder, users can open project in different IDEs according to their need, such as "Keil4_project", open \Template\Keil4_project\Project.uvproj, shown as below:

Figure 2-4. Open the project file

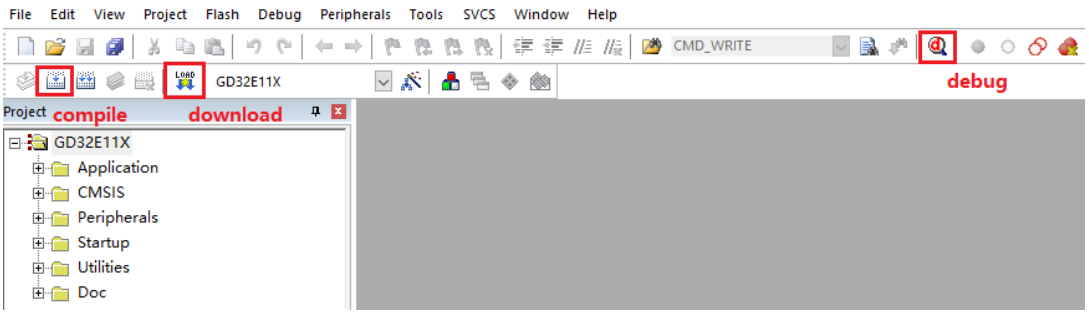


Because different module and different functions adopt different files, users should add or delete the files in project according to the copied files.

Compile-Debug-Download

First compile the project, if there is no error, then select the right jumper cap according to the description of readme, download the project to the target board, and there will be the phenomenon showed accord with the description of readme. The usage of IDE can refer to corresponding software user guide. If users are using Keil, the figure is shown as below:

Figure 2-5. Compile-debug-download



2.1.4. Utilities Folder

Utilities folder includes files about the firmware examples on evaluation board:

- LCD_Commom and Third_Party subfolders include files for USB tests;
- gd32e11x_eval.h and gd32e11x_lcd_eval.h are related header files of the evaluation board about running the firmware examples;
- gd32e11x_eval.c and gd32e11x_lcd_eval.c are related source files of the evaluation board about running the firmware examples.

Note: All the codes accord with MISRA-C:2004 standard, and will not be influenced by different software IDEs.

2.2. File descriptions of Firmware Library

The major files about the firmware library are listed and described in the table below.

Table 2-1. Function descriptions of Firmware Library

Files	Descriptions
gd32e11x_libopt.h	The header file about all the header files of peripherals. It is the only one file which is necessity to be included in the user's application, to connect the firmware library and the application.
main.c	Example of main function.
gd32e11x_it.h	Header file, including all the prototypes of interrupt service routines.
gd32e11x_it.c	Source files about interruput service routines of peripherals. User can written his own interrupt functions in this file. For the different interrupt service requests to the same interrupt vector, users can confirm the interrupt source by functions of judging interrupt flags of peripherals. The functions are included in the firmware library.
gd32e11x_xxx.h	The header file of peripheral PPP, including functions about peripheral PPP, and the variables used for functions.

gd32e11x_xxx.c	The C source file for driving peripheral PPP.
systick.h	The header file of systick.c, including prototypes of systick configuration function and delay function.
systick.c	The source file about systick configuration function and delay function.
readme.txt	Description document about how to configure and how to use the firmware example.

3. Firmware Library of Standard Peripherals

3.1. Overview of Firmware Library of Standard Peripherals

The description format of firmware functions are shown as below:

Table 3-1. Peripheral function format of Firmware Library

Function name	Name of peripheral function
Function prototype	Declaration prototype
Function descriptions	Explain the function how to work
Precondition	Requirements should meet before calling this function
The called functions	Other firmware functions called in this functin
Input parameter{in}	
Input parameter name	Description
xxxx	Description of input parameters
Output parameter{out}	
Output parameter name	Description
xxxx	Description of output parameters
Return value	
Return value type	The range of return value

3.2. ADC

The 12-bit ADC is an analog-to-digital converter using the successive approximation method. The ADC registers are listed in chapter ***3.2.1***, the ADC firmware functions are introduced in chapter ***3.2.2***.

3.2.1. Descriptions of Peripheral registers

ADC registers are listed in the table shown as below:

Table 3-2. ADC Registers

Registers	Descriptions
ADC_STAT	Status register
ADC_CTL0	Control register 0
ADC_CTL1	Control register 1
ADC_SAMPT0	Sample time register 0
ADC_SAMPT1	Sample time register 1
ADC_IOFFx (x=0..3)	Inserted channel data offset register x(x=0..3)

Registers	Descriptions
ADC_WDHT	Watchdog high threshold register
ADC_WDLT	Watchdog low threshold register
ADC_RSQ0	Regular sequence register 0
ADC_RSQ1	Regular sequence register 1
ADC_RSQ2	Regular sequence register 2
ADC_ISQ	Inserted sequence register
ADC_IDATAx	Inserted data register x(x=0..3)
ADC_RDATA	Regular data register
ADC_OVSAMPCTL	Oversample control register

3.2.2. Descriptions of Peripheral functions

ADC firmware functions are listed in the table shown as below:

Table 3-3. ADC firmware function

Function name	Function description
adc_deinit	reset ADCx peripheral
adc_mode_config	configure the ADC sync mode
adc_special_function_config	enable or disable ADC special function
adc_data_alignment_config	configure ADC data alignment
adc_enable	enable ADC interface
adc_disable	disable ADC interface
adc_calibration_enable	ADC calibration and reset calibration
adc_tempsensor_vrefint_enable	enable the temperature sensor and Vrefint channel
adc_tempsensor_vrefint_disable	disable the temperature sensor and Vrefint channel
adc_resolution_config	configure ADC resolution
adc_oversample_mode_config	configure ADC oversample mode
adc_oversample_mode_enable	enable ADC oversample mode
adc_oversample_mode_disable	disable ADC oversample mode
adc_dma_mode_enable	enable DMA request
adc_dma_mode_disable	disable DMA request
adc_discontinuous_mode_config	configure ADC discontinuous mode
adc_channel_length_config	configure the length of regular channel group or inserted channel group
adc_regular_channel_config	configure ADC regular channel
adc_inserted_channel_config	configure ADC inserted channel
adc_inserted_channel_offset_config	configure ADC inserted channel offset
adc_external_trigger_source_config	configure ADC external trigger source
adc_external_trigger_config	enable ADC external trigger
adc_software_trigger_enable	enable ADC software trigger
adc_regular_data_read	read ADC regular group data register
adc_inserted_data_read	read ADC inserted group data register

Function name	Function description
adc_sync_mode_convert_value_read	read the last ADC0 and ADC1 conversion result data in sync mode
adc_watchdog_single_channel_enable	configure ADC analog watchdog single channel
adc_watchdog_group_channel_enable	configure ADC analog watchdog group channel
adc_watchdog_disable	disable ADC analog watchdog
adc_watchdog_threshold_config	configure ADC analog watchdog threshold
adc_flag_get	get the ADC flag bits
adc_flag_clear	clear the ADC flag bits
adc_regular_software_startconv_flag_get	get the bit state of ADCx software start conversion
adc_inserted_software_startconv_flag_get	get the bit state of ADCx software inserted channel start conversion
adc_interrupt_flag_get	get the ADC interrupt bits
adc_interrupt_flag_clear	clear the ADC flag
adc_interrupt_enable	enable ADC interrupt
adc_interrupt_disable	disable ADC interrupt

adc_deinit

The description of adc_deinit is shown as below:

Table 3-4. Function adc_deinit

Function name	adc_deinit
Function prototype	void adc_deinit(uint32_t adc_periph);
Function descriptions	reset ADCx peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset ADC0 */
adc_deinit(ADC0);
```

adc_mode_config

The description of adc_mode_config is shown as below:

Table 3-5. Function adc_mode_config

Function name	adc_mode_config
Function prototype	void adc_mode_config(uint32_t mode);
Function descriptions	configure the ADCs sync mode
Precondition	-
The called functions	-
Input parameter{in}	
mode	ADC mode
ADC_MODE_FREE	all the ADC work independently
ADC_DAUL_REGULAL_PARALLEL_INSERTED_PARALLEL	ADC0 and ADC1 work in combined regular parallel + inserted parallel mode
ADC_DAUL_REGULAL_PARALLEL_INSERTED_ROTATION	ADC0 and ADC1 work in combined regular parallel + trigger rotation mode
ADC_DAUL_INSERTED_PARALLEL_REGULAL_FOLLOWUP_FAST	ADC0 and ADC1 work in combined inserted parallel + follow-up fast mode
ADC_DAUL_INSERTED_PARALLEL_REGULAL_FOLLOWUP_SLOW	ADC0 and ADC1 work in combined inserted parallel + follow-up slow mode
ADC_DAUL_INSERTED_PARALLEL	ADC0 and ADC1 work in inserted parallel mode only
ADC_DAUL_REGULAL_PARALLEL	ADC0 and ADC1 work in regular parallel mode only
ADC_DAUL_REGULAL_FOLLOWUP_FAST	ADC0 and ADC1 work in follow-up fast mode only
ADC_DAUL_REGULAL_FOLLOWUP_SLOW	ADC0 and ADC1 work in follow-up slow mode only
ADC_DAUL_INSERTED_TRRIGGER_ROTATION	ADC0 and ADC1 work in trigger rotation mode only
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the ADC sync mode */
```

adc_mode_config(ADC_MODE_FREE);

adc_special_function_config

The description of adc_special_function_config is shown as below:

Table 3-6. Function adc_special_function_config

Function name	adc_special_function_config
Function prototype	void adc_special_function_config(uint32_t adc_periph, uint32_t function, ControlStatus newvalue);
Function descriptions	enable or disable ADC special function
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
function	the function to config
ADC_SCAN_MODE	scan mode select
ADC_INSERTED_CHANNEL_AUTO	inserted channel group convert automatically
ADC_CONTINUOUS_MODE	continuous mode select
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC0 scan mode */

adc_special_function_config(ADC0, ADC_SCAN_MODE, ENABLE);

adc_data_alignment_config

The description of adc_data_alignment_config is shown as below:

Table 3-7. Function adc_data_alignment_config

Function name	adc_data_alignment_config
Function prototype	void adc_data_alignment_config(uint32_t adc_periph, uint32_t data_alignment);

Function descriptions	configure ADCx data alignment
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
data_alignment	data alignment select
ADC_DATAALIGN_RIGHT	LSB alignment
ADC_DATAALIGN_LEFT	MSB alignment
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 data alignment */
adc_data_alignment_config(ADC0, ADC_DATAALIGN_RIGHT);
```

adc_enable

The description of adc_enable is shown as below:

Table 3-8. Function adc_enable

Function name	adc_enable
Function prototype	void adc_enable(uint32_t adc_periph);
Function descriptions	enable ADCx interface
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC0 */
adc_enable(ADC0);
```

adc_disable

The description of adc_disable is shown as below:

Table 3-9. Function adc_disable

Function name	adc_disable
Function prototype	void adc_disable(uint32_t adc_periph);
Function descriptions	disable ADCx interface
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 */

adc_disable(ADC0);
```

adc_calibration_enable

The description of adc_calibration_enable is shown as below:

Table 3-10. Function adc_calibration_enable

Function name	adc_calibration_enable
Function prototype	void adc_calibration_enable(uint32_t adc_periph);
Function descriptions	ADCx calibration and reset calibration
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* ADC0 calibration and reset calibration */

adc_calibration_enable(ADC0);
```

adc_tempsensor_vrefint_enable

The description of adc_tempsensor_vrefint_enable is shown as below:

Table 3-11. Function adc_tempsensor_vrefint_enable

Function name	adc_tempsensor_vrefint_enable
Function prototype	void adc_tempsensor_vrefint_enable(void);
Function descriptions	enable the temperature sensor and Vrefint channel
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the temperature sensor and Vrefint channel */
adc_tempsensor_vrefint_enable();
```

adc_tempsensor_vrefint_disable

The description of adc_tempsensor_vrefint_disable is shown as below:

Table 3-12. Function adc_tempsensor_vrefint_disable

Function name	adc_tempsensor_vrefint_disable
Function prototype	void adc_tempsensor_vrefint_disable(void);
Function descriptions	disable the temperature sensor and Vrefint channel
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the temperature sensor and Vrefint channel */
adc_tempsensor_vrefint_disable();
```

adc_resolution_config

The description of adc_resolution_config is shown as below:

Table 3-13. Function adc_resolution_config

Function name	adc_resolution_config
Function prototype	void adc_resolution_config(uint32_t adc_periph , uint32_t resolution);
Function descriptions	configure ADC resolution
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
resolution	ADC resolution
ADC_RESOLUTION_12B	12-bit ADC resolution
ADC_RESOLUTION_10B	10-bit ADC resolution
ADC_RESOLUTION_8B	8-bit ADC resolution
ADC_RESOLUTION_6B	6-bit ADC resolution
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 data alignment */
adc_data_alignment_config(ADC0, ADC_DATAALIGN_RIGHT);
```

adc_oversample_mode_config

The description of adc_oversample_mode_config is shown as below:

Table 3-14. Function adc_oversample_mode_config

Function name	adc_oversample_mode_config
Function prototype	void adc_oversample_mode_config(uint32_t adc_periph, uint32_t mode, uint16_t shift, uint8_t ratio);
Function descriptions	configure ADC oversample mode
Precondition	-
The called functions	-

Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
mode	ADC oversampling mode
ADC_OVERSAMPLING_ALL_CONVERT	all oversampled conversions for a channel are done consecutively after a trigger
ADC_OVERSAMPLING_ONE_CONVERT	each oversampled conversion for a channel needs a trigger
Input parameter{in}	
shift	ADC oversampling shift
ADC_OVERSAMPLING_SHIFT_NONE	no oversampling shift
ADC_OVERSAMPLING_SHIFT_1B	1-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_2B	2-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_3B	3-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_4B	4-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_5B	5-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_6B	6-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_7B	7-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_8B	8-bit oversampling shift
Input parameter{in}	
ratio	ADC oversampling ratio
ADC_OVERSAMPLING_RATIO_MUL2	oversampling ratio multiple 2
ADC_OVERSAMPLING_RATIO_MUL4	oversampling ratio multiple 4
ADC_OVERSAMPLING_RATIO_MUL8	oversampling ratio multiple 8
ADC_OVERSAMPLING_RATIO_MUL16	oversampling ratio multiple 16
ADC_OVERSAMPLING_RATIO_MUL32	oversampling ratio multiple 32
ADC_OVERSAMPLING_RATIO_MUL64	oversampling ratio multiple 64

ADC_OVERSAMPLING_RATIO_MUL128	oversampling ratio multiple 128
ADC_OVERSAMPLING_RATIO_MUL256	oversampling ratio multiple 256
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC1 oversample mode: 16 times sample, 4 bits shift */
adc_oversample_mode_config(ADC1, ADC_OVERSAMPLING_ALL_CONVERT,
ADC_OVERSAMPLING_SHIFT_4B, ADC_OVERSAMPLING_RATIO_MUL16);
```

adc_oversample_mode_enable

The description of adc_oversample_mode_enable is shown as below:

Table 3-15. Function adc_oversample_mode_enable

Function name	adc_oversample_mode_enable
Function prototype	void adc_oversample_mode_enable(uint32_t adc_periph);
Function descriptions	enable ADC oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC0 oversample mode */
adc_oversample_mode_enable (ADC0);
```

adc_oversample_mode_disable

The description of adc_oversample_mode_disable is shown as below:

Table 3-16. Function adc_oversample_mode_disable

Function name	adc_oversample_mode_disable
Function prototype	void adc_oversample_mode_disable(uint32_t adc_periph);

Function descriptions	disable ADC oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 oversample mode */
adc_oversample_mode_disable (ADC0);
```

adc_dma_mode_enable

The description of adc_dma_mode_enable is shown as below:

Table 3-17. Function adc_dma_mode_enable

Function name	adc_dma_mode_enable
Function prototype	void adc_dma_mode_enable(uint32_t adc_periph);
Function descriptions	enable ADCx DMA request
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC0 DMA request */
adc_dma_mode_enable(ADC0);
```

adc_dma_mode_disable

The description of adc_dma_mode_disable is shown as below:

Table 3-18. Function adc_dma_mode_disable

Function name	adc_dma_mode_disable

Function prototype	void adc_dma_mode_disable(uint32_t adc_periph);
Function descriptions	disable ADCx DMA request
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 DMA request */
adc_dma_mode_disable(ADC0);
```

adc_discontinuous_mode_config

The description of adc_discontinuous_mode_config is shown as below:

Table 3-19. Function adc_discontinuous_mode_config

Function name	adc_discontinuous_mode_config
Function prototype	void adc_discontinuous_mode_config(uint32_t adc_periph, uint8_t adc_channel_group, uint8_t length);
Function descriptions	configure ADC discontinuous mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
ADC_CHANNEL_DISCON_DISABLE	disable discontinuous mode of regular and inserted channel
Input parameter{in}	
length	number of conversions in discontinuous mode, the number can be 1..8 for regular channel, the number has no effect for inserted channel
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* configure ADC0 regular channel group discontinuous mode */

adc_discontinuous_mode_config(ADC0, ADC_REGULAR_CHANNEL, 6);

adc_channel_length_config

The description of adc_channel_length_config is shown as below:

Table 3-20. Function adc_channel_length_config

Function name	adc_channel_length_config
Function prototype	void adc_channel_length_config(uint32_t adc_periph, uint8_t adc_channel_group, uint32_t length);
Function descriptions	configure the length of regular channel group or inserted channel group
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
length	the length of the channel, regular channel 1-16, inserted channel 1-4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the length of ADC0 regular channel */

adc_channel_length_config(ADC0, ADC_REGULAR_CHANNEL, 4);

adc_regular_channel_config

The description of adc_regular_channel_config is shown as below:

Table 3-21. Function adc_regular_channel_config

Function name	adc_regular_channel_config
Function prototype	void adc_regular_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	configure ADC regular channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
rank	the regular group sequence rank, this parameter must be between 0 to 15
Input parameter{in}	
adc_channel	the selected ADC channel
ADC_CHANNEL_x(x=0..17)	ADC Channelx (x=0..17)(x=16 and x=17 are only for ADC0)
Input parameter{in}	
sample_time	the sample time value
ADC_SAMPLETIME_1POINT5	1.5 cycles
ADC_SAMPLETIME_7POINT5	7.5 cycles
ADC_SAMPLETIME_13POINT5	13.5 cycles
ADC_SAMPLETIME_28POINT5	28.5 cycles
ADC_SAMPLETIME_41POINT5	41.5 cycles
ADC_SAMPLETIME_55POINT5	55.5 cycles
ADC_SAMPLETIME_71POINT5	71.5 cycles
ADC_SAMPLETIME_239POINT5	239.5 cycles
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 regular channel */
adc_regular_channel_config(ADC0, 1, ADC_CHANNEL_0, ADC_SAMPLETIME_7POINT5);
```

adc_inserted_channel_config

The description of adc_inserted_channel_config is shown as below:

Table 3-22. Function adc_inserted_channel_config

Function name	adc_inserted_channel_config
Function prototype	void adc_inserted_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	configure ADC inserted channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(0,1)	ADC peripheral selection
Input parameter{in}	
rank	the inserted group sequencer rank, this parameter must be between 0 to 3
Input parameter{in}	
adc_channel	the selected ADC channel
ADC_CHANNEL_x(x=0..17)	ADC Channelx (x=0..17)(x=16 and x=17 are only for ADC0)
Input parameter{in}	
sample_time	the sample time value
ADC_SAMPLETIME_1POINT5	1.5 cycles
ADC_SAMPLETIME_7POINT5	7.5 cycles
ADC_SAMPLETIME_13POINT5	13.5 cycles
ADC_SAMPLETIME_28POINT5	28.5 cycles
ADC_SAMPLETIME_41POINT5	41.5 cycles
ADC_SAMPLETIME_55POINT5	55.5 cycles
ADC_SAMPLETIME_71POINT5	71.5 cycles
ADC_SAMPLETIME_239POINT5	239.5 cycles
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC0 inserted channel */

adc_inserted_channel_config(ADC0, 1, ADC_CHANNEL_0, ADC_SAMPLETIME_7POINT5);

adc_inserted_channel_offset_config

The description of adc_inserted_channel_offset_config is shown as below:

Table 3-23. Function adc_inserted_channel_offset_config

Function name	adc_inserted_channel_offset_config
Function prototype	void adc_inserted_channel_offset_config(uint32_t adc_periph, uint8_t inserted_channel, uint16_t offset);
Function descriptions	configure ADC inserted channel offset
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
inserted_channel	insert channel select
ADC_INSERTED_CHANNEL_x(x=0..3)	inserted channel, x=0,1,2,3
Input parameter{in}	
offset	the offset data, this parameter must be between 0 to 4095
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC0 inserted channel offset */

adc_inserted_channel_offset_config(ADC0, ADC_INSERTED_CHANNEL_0, 100);

adc_external_trigger_source_config

The description of adc_external_trigger_source_config is shown as below:

Table 3-24. Function adc_external_trigger_source_config

Function name	adc_external_trigger_ source_config
Function prototype	void adc_external_trigger_source_config(uint32_t adc_periph, uint8_t adc_channel_group, uint32_t external_trigger_source);
Function descriptions	configure ADC external trigger source
Precondition	-

The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
external_trigger_source	regular or inserted group trigger source
ADC0_1_EXTTRIG_REGULAR_T0_CH0	TIMER0 CH0 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T0_CH1	TIMER0 CH1 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T0_CH2	TIMER0 CH2 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T1_CH1	TIMER1 CH1 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T2_TRGO	TIMER2 TRGO event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T3_CH3	TIMER3 CH3 event select for regular channel
ADC0_1_EXTTRIG_REGULAR_T7_TRGO	TIMER7 TRGO event select for regular channel
ADC0_1_EXTTRIG_REGULAR_EXTI_11	external interrupt line 11 for regular channel
ADC0_1_EXTTRIG_REGULAR_NONE	software trigger for regular channel
ADC0_1_EXTTRIG_INSERTED_T0_TRGO	TIMER0 TRGO event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T0_CH3	TIMER0 CH3 event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T1_TRGO	TIMER1 TRGO event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T1_CH0	TIMER1 CH0 event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T2_CH3	TIMER2 CH3 event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_T3_TRGO	TIMER3 TRGO event select for inserted channel

ADC0_1_EXTTRIG_INSERTED_EXTI_15	external interrupt line 15 for inserted channel
ADC0_1_EXTTRIG_INSERTED_T7_CH3	TIMER7 CH3 event select for inserted channel
ADC0_1_EXTTRIG_INSERTED_NONE	software trigger for inserted channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC0 regular channel external trigger source */

adc_external_trigger_source_config(ADC0,ADC_REGULAR_CHANNEL, ADC0_1_EXTTRIG_REGULAR_T0_CH0);

adc_external_trigger_config

The description of adc_external_trigger_config is shown as below:

Table 3-25. Function adc_external_trigger_config

Function name	adc_external_trigger_config
Function prototype	void adc_external_trigger_config(uint32_t adc_periph, uint8_t adc_channel_group, ControlStatus newvalue);
Function descriptions	configure ADC external trigger
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable ADC0 inserted channel group external trigger */

adc_external_trigger_config(ADC0, ADC_INSERTED_CHANNEL_0, ENABLE);

adc_software_trigger_enable

The description of adc_software_trigger_enable is shown as below:

Table 3-26. Function adc_software_trigger_enable

Function name	adc_software_trigger_enable
Function prototype	void adc_software_trigger_enable(uint32_t adc_periph, uint8_t adc_channel_group);
Function descriptions	enable ADC software trigger
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC0 regular channel group software trigger */

adc_software_trigger_enable(ADC0, ADC_REGULAR_CHANNEL);

adc_regular_data_read

The description of adc_regular_data_read is shown as below:

Table 3-27. Function adc_regular_data_read

Function name	adc_regular_data_read
Function prototype	uint16_t adc_regular_data_read(uint32_t adc_periph);
Function descriptions	read ADC regular group data register

Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	ADC conversion value (0-0xFFFF)

Example:

```
/* read ADC0 regular group data register */
uint16_t adc_value = 0;
adc_value = adc_regular_data_read(ADC0);
```

adc_inserted_data_read

The description of adc_inserted_data_read is shown as below:

Table 3-28. Function adc_inserted_data_read

Function name	adc_inserted_data_read
Function prototype	uint16_t adc_inserted_data_read(uint32_t adc_periph, uint8_t inserted_channel);
Function descriptions	read ADC inserted group data register
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
inserted_channel	insert channel select
ADC_INSERTED_CHANNEL_x(x=0..3)	inserted Channelx, x=0,1,2,3
Output parameter{out}	
-	-
Return value	
uint16_t	ADC conversion value (0-0xFFFF)

Example:

```
/* read ADC0 inserted group data register */
uint16_t adc_value = 0;
adc_value = adc_inserted_data_read (ADC0, ADC_INSERTED_CHANNEL_0);
```

adc_sync_mode_convert_value_read

The description of adc_sync_mode_convert_value_read is shown as below:

Table 3-29. Function adc_sync_mode_convert_value_read

Function name	adc_sync_mode_convert_value_read
Function prototype	uint32_t adc_sync_mode_convert_value_read(void);
Function descriptions	read the last ADC0 and ADC1 conversion result data in sync mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	ADC conversion value (0-0xFFFFFFFF)

Example:

```
/* read the last ADC0 and ADC1 conversion result data in sync mode */
uint32_t adc_value = 0;
adc_value = adc_sync_mode_convert_value_read ();
```

adc_watchdog_single_channel_enable

The description of adc_watchdog_single_channel_enable is shown as below:

Table 3-30. Function adc_watchdog_single_channel_enable

Function name	adc_watchdog_single_channel_enable
Function prototype	void adc_watchdog_single_channel_enable(uint32_t adc_periph, uint8_t adc_channel);
Function descriptions	configure ADC analog watchdog single channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel	the selected ADC channel
ADC_CHANNEL_x(x=0..17)	ADC Channelx(x=0..17) (x=16 and x=17 are only for ADC0)
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* configure ADC0 analog watchdog single channel */

adc_watchdog_single_channel_enable(ADC0, ADC_CHANNEL_1);

adc_watchdog_group_channel_enable

The description of adc_watchdog_group_channel_enable is shown as below:

Table 3-31. Function adc_watchdog_group_channel_enable

Function name	adc_watchdog_group_channel_enable
Function prototype	void adc_watchdog_group_channel_enable(uint32_t adc_periph, uint8_t adc_channel_group);
Function descriptions	configure ADC analog watchdog group channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	the channel group use analog watchdog
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
ADC_REGULAR_INSERTED_CHANNEL	both regular and inserted group
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure ADC0 analog watchdog group channel */

adc_watchdog_group_channel_enable(ADC0, ADC_REGULAR_CHANNEL);

adc_watchdog_disable

The description of adc_watchdog_disable is shown as below:

Table 3-32. Function adc_watchdog_disable

Function name	adc_watchdog_disable

Function prototype	void adc_watchdog_disable(uint32_t adc_periph);
Function descriptions	disable ADC analog watchdog
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 analog watchdog */
adc_watchdog_disable(ADC0);
```

adc_watchdog_threshold_config

The description of adc_watchdog_threshold_config is shown as below:

Table 3-33. Function adc_watchdog_threshold_config

Function name	adc_watchdog_threshold_config
Function prototype	void adc_watchdog_threshold_config(uint32_t adc_periph, uint16_t low_threshold, uint16_t high_threshold);
Function descriptions	configure ADC analog watchdog threshold
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
low_threshold	analog watchdog low threshold, 0..4095
Input parameter{in}	
high_threshold	analog watchdog high threshold, 0..4095
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 analog watchdog threshold */
adc_watchdog_threshold_config(ADC0, 0x0400, 0x0A00);
```

adc_flag_get

The description of adc_flag_get is shown as below:

Table 3-34. Function adc_flag_get

Function name	adc_flag_get
Function prototype	FlagStatus adc_flag_get(uint32_t adc_periph , uint32_t adc_flag);
Function descriptions	get the ADC flag bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_flag	the adc flag bits
ADC_FLAG_WDE	analog watchdog event flag
ADC_FLAG_EOC	end of group conversion flag
ADC_FLAG_EOIC	end of inserted group conversion flag
ADC_FLAG_STIC	start flag of inserted channel group
ADC_FLAG_STRC	start flag of regular channel group
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ADC0 analog watchdog flag bits*/
FlagStatus flag_value;
flag_value = adc_flag_get(ADC0, ADC_FLAG_WDE);
```

adc_flag_clear

The description of adc_flag_clear is shown as below:

Table 3-35. Function adc_flag_clear

Function name	adc_flag_clear
Function prototype	void adc_flag_clear(uint32_t adc_periph, uint32_t adc_flag);
Function descriptions	clear the ADC flag bits
Precondition	-
The called functions	-

Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_flag	the adc flag bits
ADC_FLAG_WDE	analog watchdog event flag
ADC_FLAG_EOC	end of group conversion flag
ADC_FLAG_EOIC	end of inserted group conversion flag
ADC_FLAG_STIC	start flag of inserted channel group
ADC_FLAG_STRC	start flag of regular channel group
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the ADC0 analog watchdog flag bits*/
adc_flag_clear(ADC0, ADC_FLAG_WDE);
```

adc_regular_software_startconv_flag_get

The description of adc_regular_software_startconv_flag_get is shown as below:

Table 3-36. Function adc_regular_software_startconv_flag_get

Function name	adc_regular_software_startconv_flag_get
Function prototype	FlagStatus adc_regular_software_startconv_flag_get(uint32_t adc_periph);
Function descriptions	get the bit state of ADCx software regular channel start conversion
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the bit state of ADC0 software regular channel start conversion */

FlagStatus flag_value;

flag_value = adc_regular_software_startconv_flag_get(ADC0);

adc_inserted_software_startconv_flag_get

The description of adc_inserted_software_startconv_flag_get is shown as below:

Table 3-37. Function adc_inserted_software_startconv_flag_get

Function name	adc_inserted_software_startconv_flag_get
Function prototype	FlagStatus adc_inserted_software_startconv_flag_get(uint32_t adc_periph);
Function descriptions	get the bit state of ADCx software inserted channel start conversion
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get the bit state of ADC0 software inserted channel start conversion */

FlagStatus flag_value;

flag_value = adc_inserted_software_startconv_flag_get(ADC0);

adc_interrupt_flag_get

The description of adc_interrupt_flag_get is shown as below:

Table 3-38. Function adc_interrupt_flag_get

Function name	adc_interrupt_flag_get
Function prototype	FlagStatus adc_interrupt_flag_get(uint32_t adc_periph, uint32_t adc_interrupt);
Function descriptions	get the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection

Input parameter{in}	
adc_interrupt	the adc interrupt bits
ADC_INT_WDE	analog watchdog interrupt
ADC_INT_EOC	end of group conversion interrupt
ADC_INT_EOIC	end of inserted group conversion interrupt
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ADC0 analog watchdog interrupt bits*/
FlagStatus flag_value;
flag_value = adc_interrupt_flag_get(ADC0, ADC_INT_WDE);
```

adc_interrupt_flag_clear

The description of adc_interrupt_flag_clear is shown as below:

Table 3-39. Function adc_interrupt_flag_clear

Function name	adc_interrupt_flag_clear
Function prototype	void adc_interrupt_flag_clear(uint32_t adc_periph, uint32_t adc_interrupt);
Function descriptions	clear the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_interrupt	the adc interrupt bits
ADC_INT_WDE	analog watchdog interrupt
ADC_INT_EOC	end of group conversion interrupt
ADC_INT_EOIC	end of inserted group conversion interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the ADC0 analog watchdog interrupt bits*/
adc_interrupt_flag_clear(ADC0, ADC_INT_WDE);
```

adc_interrupt_enable

The description of adc_interrupt_enable is shown as below:

Table 3-40. Function adc_interrupt _enable

Function name	adc_interrupt_enable
Function prototype	void adc_interrupt_enable(uint32_t adc_periph, uint32_t adc_interrupt);
Function descriptions	enable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_interrupt	the adc interrupt
ADC_INT_WDE	analog watchdog interrupt
ADC_INT_EOC	end of group conversion interrupt
ADC_INT_EOIC	end of inserted group conversion interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable ADC0 analog watchdog interrupt */

adc_interrupt_enable(ADC0, ADC_INT_WDE);

adc_interrupt_disable

The description of adc_interrupt_disable is shown as below:

Table 3-41. Function adc_interrupt_disable

Function name	adc_interrupt_disable
Function prototype	void adc_interrupt_disable(uint32_t adc_periph , uint32_t adc_interrupt);
Function descriptions	Disable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_interrupt	the adc interrupt
ADC_INT_WDE	analog watchdog interrupt
ADC_INT_EOC	end of group conversion interrupt

ADC_INT_EOIC	end of inserted group conversion interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 interrupt */
adc_interrupt_disable(ADC0, ADC_INT_WDE);
```

3.3. BKP

The Backup registers are located in the Backup domain that remains powered-on by V_{BAT} even if V_{DD} power is shut down, they are forty two 16-bit (84 bytes) registers for data protection of user application data, and the wake-up action from Standby mode or system reset do not affect these registers. The BKP registers are listed in chapter *3.3.1*, the BKP firmware functions are introduced in chapter *3.3.2*.

3.3.1. Descriptions of Peripheral registers

BKP registers are listed in the table shown as below:

Table 3-42. BKP Registers

Registers	Descriptions
BKP_DATAx (x = 0..41)	Backup data register
BKP_OCTL	RTC signal output control register
BKP_TPCTL	Tamper pin control register
BKP_TPCS	Tamper control and status register

3.3.2. Descriptions of Peripheral functions

BKP firmware functions are listed in the table shown as below:

Table 3-43. BKP firmware function

Function name	Function description
bkp_deinit	reset data registers
bkp_data_write	write data register
bkp_data_read	read data register
bkp_rtc_calibration_output_enable	enable RTC clock calibration output
bkp_rtc_calibration_output_disable	disable RTC clock calibration output
bkp_rtc_signal_output_enable	enable RTC alarm or second signal output
bkp_rtc_signal_output_disable	disable RTC alarm or second signal output

Function name	Function description
bkp_rtc_output_select	select RTC output, the RTC output can be select as alarm pulse or second pulse
bkp_rtc_clock_output_select	select RTC clock output
bkp_rtc_clock_calibration_direction_select	select RTC clock calibration direction
bkp_rtc_calibration_value_set	set RTC clock calibration value
bkp_tamper_detection_enable	enable tamper detection
bkp_tamper_detection_disable	disable tamper detection
bkp_tamper_active_level_set	set tamper pin active level
bkp_interrupt_enable	enable tamper interrupt
bkp_interrupt_disable	disable tamper interrupt
bkp_flag_get	get bkp flag state
bkp_flag_clear	clear bkp flag state
bkp_interrupt_flag_get	get bkp interrupt flag state
bkp_interrupt_flag_clear	clear bkp interrupt flag state

bkp_deinit

The description of bkp_deinit is shown as below:

Table 3-44. Function bkp_deinit

Function name	bkp_deinit
Function prototype	void bkp_deinit(void);
Function descriptions	reset data registers
Precondition	-
The called functions	rcu_bkp_reset_enable / rcu_bkp_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset BKP registers */

bkp_deinit();
```

bkp_data_write

The description of bkp_data_write is shown as below:

Table 3-45. Function bkp_data_write

Function name	bkp_data_write

Function prototype	void bkp_data_write(bkp_data_register_enum register_number, uint16_t data);
Function descriptions	write data register
Precondition	-
The called functions	-
Input parameter{in}	
register_number	refer to bkp_data_register_enum
BKP_DATA_x(x = 0..41)	bkp data register number x
Input parameter{in}	
data	the data to be write in BKP data register
0-0xffff	data value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write BKP data register */
bkp_data_write(BKP_DATA_0, 0x1226);
```

bkp_data_read

The description of bkp_data_read is shown as below:

Table 3-46. Function bkp_data_read

Function name	bkp_data_read
Function prototype	uint16_t bkp_data_read(bkp_data_register_enum register_number);
Function descriptions	read data register
Precondition	-
The called functions	-
Input parameter{in}	
register_number	refer to bkp_data_register_enum
BKP_DATA_x(x = 0..41)	bkp data register number x
Output parameter{out}	
-	-
Return value	
uint16_t	0-0xffff

Example:

```
/* read BKP data register */
uint16_t data;
```

data = bkp_data_read(BKP_DATA_0);

bkp_rtc_calibration_output_enable

The description of bkp_rtc_calibration_output_enable is shown as below:

Table 3-47. Function bkp_rtc_calibration_output_enable

Function name	bkp_rtc_calibration_output_enable
Function prototype	void bkp_rtc_calibration_output_enable(void);
Function descriptions	enable RTC clock calibration output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC clock calibration output */

bkp_rtc_calibration_output_enable();
```

bkp_rtc_calibration_output_disable

The description of bkp_rtc_calibration_output_disable is shown as below:

Table 3-48. Function bkp_rtc_calibration_output_disable

Function name	bkp_rtc_calibration_output_disable
Function prototype	void bkp_rtc_calibration_output_disable(void);
Function descriptions	disable RTC clock calibration output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC clock calibration output */

bkp_rtc_calibration_output_disable();
```

bkp_rtc_signal_output_enable

The description of bkp_rtc_signal_output_enable is shown as below:

Table 3-49. Function bkp_rtc_signal_output_enable

Function name	bkp_rtc_signal_output_enable
Function prototype	void bkp_rtc_signal_output_enable(void);
Function descriptions	enable RTC alarm or second signal output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC alarm or second signal output */
bkp_rtc_signal_output_enable();
```

bkp_rtc_signal_output_disable

The description of bkp_rtc_signal_output_disable is shown as below:

Table 3-50. Function bkp_rtc_signal_output_disable

Function name	bkp_rtc_signal_output_disable
Function prototype	void bkp_rtc_signal_output_disable(void);
Function descriptions	disable RTC alarm or second signal output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC alarm or second signal output */
bkp_rtc_signal_output_disable();
```

bkp_rtc_output_select

The description of bkp_rtc_output_select is shown as below:

Table 3-51. Function bkp_rtc_output_select

Function name	bkp_rtc_output_select
Function prototype	void bkp_rtc_output_select(uint16_t outputsel);
Function descriptions	select RTC output, the RTC output can be select as alarm pulse or second pulse
Precondition	-
The called functions	-
Input parameter{in}	
outputsel	RTC output selection
RTC_OUTPUT_ALARM_PULSE	RTC alarm pulse is selected as the RTC output
RTC_OUTPUT_SECOND_PULSE	RTC second pulse is selected as the RTC output
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select RTC output alarm signal output */
bkp_rtc_output_select(RTC_OUTPUT_ALARM_PULSE);
```

bkp_rtc_clock_output_select

The description of bkp_rtc_clock_output_select is shown as below:

Table 3-52. Function bkp_rtc_clock_output_select

Function name	bkp_rtc_clock_output_select
Function prototype	void bkp_rtc_clock_output_select(uint16_t clocksel);
Function descriptions	select RTC clock output, the RTC clock output can be select as divided 64 or no division
Precondition	-
The called functions	-
Input parameter{in}	
clocksel	RTC clock output selection
RTC_CLOCK_DIV_64	RTC clock divided 64 is selected as the RTC clock output
RTC_CLOCK_DIV_1	RTC clock is selected as the RTC clock output
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* select RTC clock devided 64 to output */

bkp_rtc_clock_output_select(RTC_CLOCK_DIV_64);

bkp_rtc_clock_calibration_direction_select

The description of bkp_rtc_clock_calibration_direction_select is shown as below:

Table 3-53. Function bkp_rtc_clock_calibration_direction_select

Function name	bkp_rtc_clock_calibration_direction_select
Function prototype	void bkp_rtc_clock_calibration_direction_select(uint16_t direction);
Function descriptions	select RTC clock calibration direction, the RTC clock calibration direction can be select as slowed down or speed up
Precondition	-
The called functions	-
Input parameter{in}	
direction	RTC clock calibration direction
RTC_CLOCK_SLOW_DOWN	RTC clock slow down
RTC_CLOCK_SPEED_UP	RTC clock speed up
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set RTC clock slowed down */

bkp_rtc_clock_calibration_direction_select(RTC_CLOCK_SLOWED_DOWN);

bkp_rtc_calibration_value_set

The description of bkp_rtc_calibration_value_set is shown as below:

Table 3-54. Function bkp_rtc_calibration_value_set

Function name	bkp_rtc_calibration_value_set
Function prototype	void bkp_rtc_calibration_value_set(uint8_t value);
Function descriptions	set RTC clock calibration value
Precondition	-
The called functions	-
Input parameter{in}	

value	RTC clock calibration value
0x00 - 0x7F	value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set RTC clock calibration value */
bkp_rtc_calibration_value_set(0x7f);
```

bkp_tamper_detection_enable

The description of bkp_tamper_detection_enable is shown as below:

Table 3-55. Function bkp_tamper_detection_enable

Function name	bkp_tamper_detection_enable
Function prototype	void bkp_tamper_detection_enable(void);
Function descriptions	enable tamper detection
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable tamper pin detection */
bkp_tamper_detection_enable();
```

bkp_tamper_detection_disable

The description of bkp_tamper_detection_disable is shown as below:

Table 3-56. Function bkp_tamper_detection_disable

Function name	bkp_tamper_detection_disable
Function prototype	void bkp_tamper_detection_disable(void);
Function descriptions	disable tamper detection
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable tamper pin detection */
bkp_tamper_detection_disable();
```

bkp_tamper_active_level_set

The description of bkp_tamper_active_level_set is shown as below:

Table 3-57. Function bkp_tamper_active_level_set

Function name	bkp_tamper_active_level_set
Function prototype	void bkp_tamper_active_level_set(uint16_t level);
Function descriptions	set tamper pin active level
Precondition	-
The called functions	-
Input parameter{in}	
level	tamper pin active level
TAMPER_PIN_ACTIVE_HIGH	the tamper pin is active high
TAMPER_PIN_ACTIVE_LOW	the tamper pin is active low
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set tamper pin active level high */
bkp_tamper_active_level_set(TAMPER_PIN_ACTIVE_HIGH);
```

bkp_interrupt_enable

The description of bkp_interrupt_enable is shown as below:

Table 3-58. Function bkp_interrupt_enable

Function name	bkp_interrupt_enable
Function prototype	void bkp_interrupt_enable(void);
Function descriptions	enable tamper interrupt

Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable tamper pin interrupt */
bkp_interrupt_enable();
```

bkp_interrupt_disable

The description of bkp_interrupt_disable is shown as below:

Table 3-59. Function bkp_interrupt_disable

Function name	bkp_interrupt_disable
Function prototype	void bkp_interrupt_disable(void);
Function descriptions	disable tamper interrupt
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable tamper pin interrupt */
bkp_interrupt_disable();
```

bkp_flag_get

The description of bkp_flag_get is shown as below:

Table 3-60. Function bkp_flag_get

Function name	bkp_flag_get
Function prototype	FlagStatus bkp_flag_get(void);
Function descriptions	get bkp flag state
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get BKP flag state */
FlagStatus status;
status = bkp_flag_get();
```

bkp_flag_clear

The description of bkp_flag_clear is shown as below:

Table 3-61. Function bkp_flag_clear

Function name	bkp_flag_clear
Function prototype	void bkp_flag_clear(void);
Function descriptions	clear bkp flag state
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear BKP flag state */
bkp_flag_clear();
```

bkp_interrupt_flag_get

The description of bkp_interrupt_flag_get is shown as below:

Table 3-62. Function bkp_interrupt_flag_get

Function name	bkp_interrupt_flag_get
Function prototype	FlagStatus bkp_interrupt_flag_get(void);
Function descriptions	get bkp interrupt flag state
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get BKP interrupt flag state */
bkp_interrupt_flag_get ();
```

bkp_interrupt_flag_clear

The description of bkp_interrupt_flag_clear is shown as below:

Table 3-63. Function bkp_interrupt_flag_clear

Function name	bkp_interrupt_flag_clear
Function prototype	void bkp_interrupt_flag_clear(void);
Function descriptions	clear bkp interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear BKP interrupt flag state */
bkp_interrupt_flag_clear();
```

3.4. CRC

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. The CRC registers are listed in chapter *3.4.1*, the CRC firmware functions are introduced in chapter *3.4.2*.

3.4.1. Descriptions of Peripheral registers

CRC registers are listed in the table shown as below:

Table 3-64. CRC Registers

Registers	Descriptions
CRC_DATA	CRC data register
CRC_FDATA	CRC free data register
CRC_CTL	CRC control register

3.4.2. Descriptions of Peripheral functions

CRC firmware functions are listed in the table shown as below:

Table 3-65. CRC firmware function

Function name	Function description
crc_deinit	deinit CRC calculation unit
crc_data_register_reset	reset data register to the initializaiton value of data register
crc_data_register_read	read the value of the data register
crc_free_data_register_read	read the value of the free data register
crc_free_data_register_write	write data to the free data register
crc_single_data_calculate	calculate the CRC value of a 32-bit data
crc_block_data_calculate	calculate the CRC value of an array of 32-bit values

crc_deinit

The description of crc_deinit is shown as below:

Table 3-66. Function crc_deinit

Function name	crc_deinit
Function prototype	void crc_deinit(void);
Function descriptions	deinit CRC calculation unit
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset crc */

crc_deinit();
```

crc_data_register_reset

The description of crc_data_register_reset is shown as below:

Table 3-67. Function crc_data_register_reset

Function name	crc_data_register_reset
Function prototype	void crc_data_register_reset(void);
Function descriptions	reset data register to the initializaiton value of data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset crc data register */
crc_data_register_reset ();
```

crc_data_register_read

The description of crc_data_register_read is shown as below:

Table 3-68. Function crc_data_register_read

Function name	crc_data_register_read
Function prototype	uint32_t crc_data_register_read(void);
Function descriptions	read the value of the data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit value of the data register (0-0xFFFFFFFF)

Example:

```
/* read crc data register */
uint32_t crc_value = 0;
crc_value = crc_data_register_read();
```

crc_free_data_register_read

The description of crc_free_data_register_read is shown as below:

Table 3-69. Function crc_free_data_register_read

Function name	crc_free_data_register_read
Function prototype	uint8_t crc_free_data_register_read(void);
Function descriptions	read the value of the free data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	8-bit value of the free data register (0-0xFF)

Example:

```
/* read crc free data register */
uint8_t crc_value = 0;
crc_value = crc_free_data_register_read();
```

crc_free_data_register_write

The description of crc_free_data_register_write is shown as below:

Table 3-70. Function crc_free_data_register_write

Function name	crc_free_data_register_write
Function prototype	void crc_free_data_register_write(uint8_t free_data);
Function descriptions	write data to the free data register
Precondition	-
The called functions	-
Input parameter{in}	
free_data	specify 8-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write the free data register */
crc_free_data_register_write(0x11);
```

crc_single_data_calculate

The description of crc_single_data_calculate is shown as below:

Table 3-71. Function crc_single_data_calculate

Function name	crc_single_data_calculate
Function prototype	uint32_t crc_single_data_calculate(uint32_t sdata);
Function descriptions	calculate the CRC value of a 32-bit data
Precondition	-
The called functions	-
Input parameter{in}	
sdata	specify 32-bit data
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit CRC calculate value (0-0xFFFFFFFF)

Example:

```
/* CRC calculate a 32-bit data */
uint32_t val = 0, valcrc = 0;
val = (uint32_t)0xabcd1234;
rcu_periph_clock_enable(RCU_CRC);
valcrc = crc_single_data_calculate(val);
```

crc_block_data_calculate

The description of crc_block_data_calculate is shown as below:

Table 3-72. Function crc_block_data_calculate

Function name	crc_block_data_calculate
Function prototype	uint32_t crc_block_data_calculate(uint32_t array[], uint32_t size);
Function descriptions	calculate the CRC value of an array of 32-bit values
Precondition	-
The called functions	-
Input parameter{in}	
array	pointer to an array of 32 bit data words
Input parameter{in}	
size	size of the array
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit CRC calculate value (0-0xFFFFFFFF)

Example:

```
/* CRC calculate a 32-bit data array */
```

```
/* CRC calculate a 32-bit data array */

#define BUFFER_SIZE       6

uint32_t valcrc = 0;

static const uint32_t data_buffer[BUFFER_SIZE] = {

0x00001111, 0x00002222, 0x00003333, 0x00004444, 0x00005555, 0x00006666};

rcu_periph_clock_enable(RCU_CRC);

valcrc = crc_block_data_calculate((uint32_t *) data_buffer, BUFFER_SIZE);
```

3.5. CTC

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.The CTC registers are listed in chapter *3.5.1*, the CTC firmware functions are introduced in chapter *3.5.2*

3.5.1. Descriptions of Peripheral registers

CTC registers are listed in the table shown as below:

Table 3-73. CTC Registers

Registers	Descriptions
CTC_CTL0	CTC control register 0
CTC_CTL1	CTC control register 1
CTC_STAT	CTC status register
CTC_INTC	CTC Interrupt clear register

3.5.2. Descriptions of Peripheral functions

CTC registers are listed in the table shown as below:

Table 3-74. CTC firmware function

Function name	Function description
ctc_deinit	reset CTC clock trim controller
ctc_counter_enable	enable CTC trim counter
ctc_counter_disable	disable CTC trim counter
ctc_irc48m_trim_value_config	configure the IRC48M trim value
ctc_software_refsource_pulse_generate	generate software reference source sync pulse
ctc_hardware_trim_mode_config	configure hardware automatically trim mode
ctc_refsource_polarity_config	configure reference signal source polarity

Function name	Function description
ctc_refsource_signal_select	select reference signal source
ctc_refsource_prescaler_config	configure reference signal source prescaler
ctc_clock_limit_value_config	configure clock trim base limit value
ctc_counter_reload_value_config	configure CTC counter reload value
ctc_counter_capture_value_read	read CTC counter capture value when reference sync pulse occurred
ctc_counter_direction_read	read CTC trim counter direction when reference sync pulse occurred
ctc_counter_reload_value_read	read CTC counter reload value
ctc_irc48m_trim_value_read	read the IRC48M trim value
ctc_interrupt_enable	enable the CTC interrupt
ctc_interrupt_disable	disable the CTC interrupt
ctc_interrupt_flag_get	get CTC interrupt flag
ctc_interrupt_flag_clear	clear CTC interrupt flag
ctc_flag_get	get CTC flag
ctc_flag_clear	clear CTC flag

ctc_deinit

The description of ctc_deinit is shown as below:

Table 3-75. Function ctc_deinit

Function name	ctc_deinit
Function prototype	void ctc_deinit (void)
Function descriptions	Reset CTC peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset CTC */
ctc_deinit();
```

ctc_counter_enable

The description of ctc_counter_enable is shown as below:

Table 3-76. Function ctc_counter_enable

Function name	ctc_counter_enable
Function prototype	void ctc_counter_enable (void);
Function descriptions	enable CTC counter
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CTC trim counter*/

ctc_counter_enable ();
```

ctc_counter_disable

The description of ctc_counter_disable is shown as below:

Table 3-77. Function ctc_counter_disable

Function name	ctc_counter_disable
Function prototype	void ctc_counter_disable (void);
Function descriptions	disable CTC counter
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CTC trim counter */

ctc_counter_disable ();
```

ctc_irc48m_trim_value_config

The description of ctc_irc48m_trim_value_config is shown as below:

Table 3-78. Function ctc_irc48m_trim_value_config

Function name	ctc_irc48m_trim_value_config
Function prototype	void ctc_irc48m_trim_value_config(uint8_t trim_value);
Function descriptions	configure the IRC48M trim value
Precondition	-
The called functions	-
Input parameter{in}	
trim_value	0~63
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* IRC48M trim value configuration */

ctc_irc48m_trim_value_config (0x01);
```

ctc_software_refsource_pulse_generate

The description of ctc_software_refsource_pulse_generate is shown as below:

Table 3-79. Function ctc_software_refsource_pulse_generate

Function name	ctc_software_refsource_pulse_generate
Function prototype	void ctc_software_refsource_pulse_generate (void)
Function descriptions	generate software reference source sync pulse
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* generate reference source sync pulse */

ctc_software_refsource_pulse_generate ();
```

ctc_hardware_trim_mode_config

The description of ctc_hardware_trim_mode_config is shown as below:

Table 3-80. Function ctc_hardware_trim_mode_config

Function name	ctc_hardware_trim_mode_config
Function prototype	void ctc_hardware_trim_mode_config(uint32_t hardmode);
Function descriptions	configure hardware automatically trim mode
Precondition	-
The called functions	-
Input parameter{in}	
hardmode	hardware automatically trim mode enable or disable
CTC_HARDWARE_TRIM_MODE_ENABLE	hardware automatically trim mode enable
CTC_HARDWARE_TRIM_MODE_DISABLE	hardware automatically trim mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CTC hardware trim */
ctc_hardware_trim_mode_config (CTC_HARDWARE_TRIM_MODE_ENABLE);
```

ctc_refsource_polarity_config

The description of ctc_refsource_polarity_config is shown as below:

Table 3-81. Function ctc_refsource_polarity_config

Function name	ctc_refsource_polarity_config
Function prototype	void ctc_refsource_polarity_config(uint32_t polarity);
Function descriptions	configure reference signal source polarity
Precondition	-
The called functions	-
Input parameter{in}	
polarity	reference signal source polarity
CTC_REFSOURCE_POLARITY_FALLING	reference signal source polarity is falling edge
CTC_REFSOURCE_POLARITY_RISING	reference signal source polarity is rising edge
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set reference source polarity */

ctc_refsource_polarity_config (CTC_REFSOURCE_POLARITY_RISING);

ctc_refsource_signal_select

The description of ctc_refsource_signal_select is shown as below:

Table 3-82. Function ctc_refsource_signal_select

Function name	ctc_refsource_signal_select
Function prototype	void ctc_refsource_signal_select(uint32_t refs);
Function descriptions	select reference signal source
Precondition	-
The called functions	-
Input parameter{in}	
refs	reference signal source
CTC_REFSOURCE_GPIO	GPIO is selected
CTC_REFSOURCE_LXTAL	LXTAL is selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reference signal selection */

ctc_refsource_signal_select (CTC_REFSOURCE_LXTAL);

ctc_refsource_prescaler_config

The description of ctc_refsource_prescaler_config is shown as below:

Table 3-83. Function ctc_refsource_prescaler_config

Function name	ctc_refsource_prescaler_config
Function prototype	void ctc_refsource_prescaler_config(uint32_t prescaler);
Function descriptions	configure reference signal source prescaler
Precondition	-
The called functions	-
Input parameter{in}	
prescaler	Prescaler factor
CTC_REFSOURCE_PSC_OFF	reference signal not divided
CTC_REFSOURCE_P	reference signal divided by 2

SC_DIV2	
CTC_REFSOURCE_PSC_DIV4	reference signal divided by 4
CTC_REFSOURCE_PSC_DIV8	reference signal divided by 8
CTC_REFSOURCE_PSC_DIV16	reference signal divided by 16
CTC_REFSOURCE_PSC_DIV32	reference signal divided by 32
CTC_REFSOURCE_PSC_DIV64	reference signal divided by 64
CTC_REFSOURCE_PSC_DIV128	reference signal divided by 128
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure reference signal source prescaler */
ctc_refsource_prescaler_config(CTC_REFSOURCE_PSC_DIV2);
```

ctc_clock_limit_value_config

The description of ctc_clock_limit_value_config is shown as below:

Table 3-84. Function ctc_clock_limit_value_config

Function name	ctc_clock_limit_value_config
Function prototype	void ctc_clock_limit_value_config(uint8_t limit_value);
Function descriptions	configure clock trim base limit value
Precondition	-
The called functions	-
Input parameter{in}	
limit_value	0x00 - 0xFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure clock trim base limit value */
ctc_clock_limit_value_config (0x1F);
```

ctc_counter_reload_value_config

The description of ctc_counter_reload_value_config is shown as below:

Table 3-85. Function ctc_counter_reload_value_config

Function name	ctc_counter_reload_value_config
Function prototype	void ctc_counter_reload_value_config(uint16_t reload_value);
Function descriptions	configure CTC counter reload value
Precondition	-
The called functions	-
Input parameter{in}	
reload_value	0x0000 - 0xFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CTC counter reload value */
ctc_counter_reload_value_config (0x00FF);
```

ctc_counter_capture_value_read

The description of ctc_counter_capture_value_read is shown as below:

Table 3-86. Function ctc_counter_capture_value_read

Function name	ctc_counter_capture_value_read
Function prototype	uint16_t ctc_counter_capture_value_read(void);
Function descriptions	read CTC counter capture value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	读取计数器捕获值(0x0000 - 0xFFFF)

Example:

```
/* read CTC counter capture value */
uint16_t ctc_value = 0;
ctc_value = ctc_counter_capture_value_read ();
```

ctc_counter_direction_read

The description of ctc_counter_direction_read is shown as below:

Table 3-87. Function ctc_counter_direction_read

Function name	ctc_counter_direction_read
Function prototype	FlagStatus ctc_counter_direction_read(void);
Function descriptions	read CTC trim counter direction
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET(向下计数) / RESET(向上计数)

Example:

```
/* read ctc counter direction */
FlagStatus ctc_direction = SET;
ctc_direction = ctc_counter_direction_read ();
```

ctc_counter_reload_value_read

The description of ctc_counter_reload_value_read is shown as below:

Table 3-88. Function ctc_counter_reload_value_read

Function name	ctc_counter_reload_value_read
Function prototype	uint16_t ctc_counter_reload_value_read(void);
Function descriptions	read CTC counter reload value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	读取计数器重载值的16位数据 (0x0000 - 0xFFFF)

Example:

```
/* read CTC counter reload value */
uint16_t ctc_reload_value = 0;
```

ctc_reload_value = ctc_counter_reload_value_read ();

ctc_irc48m_trim_value_read

The description of ctc_irc48m_trim_value_read is shown as below:

Table 3-89. Function ctc_irc48m_trim_value_read

Function name	ctc_irc48m_trim_value_read
Function prototype	uint8_t ctc_irc48m_trim_value_read(void);
Function descriptions	read the IRC48M trim value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	**-**
Return value	
uint8_t	6位IRC48M校准值 (0-63)

Example:

```
/* read the IRC48M trim value */
uint8_t ctc_trim_value = 0;
ctc_trim_value = ctc_irc48m_trim_value_read ();
```

ctc_interrupt_enable

The description of ctc_interrupt_enable is shown as below:

Table 3-90. Function ctc_interrupt_enable

Function name	ctc_interrupt_enable
Function prototype	void ctc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable the CTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	CTC interrupt
CTC_INT_CKOK	clock trim OK interrupt
CTC_INT_CKWARN	clock trim warning interrupt
CTC_INT_ERR	error interrupt
CTC_INT_EREF	expect reference interrupt
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable CTC clock trim OK interrupt */

ctc_interrupt_enable (CTC_INT_CKOK);

ctc_interrupt_disable

The description of ctc_interrupt_disable is shown as below:

Table 3-91. Function ctc_interrupt_disable

Function name	ctc_interrupt_disable
Function prototype	void ctc_interrupt_disable(uint32_t interrupt);
Function descriptions	disable the CTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	CTC interrupt
CTC_INT_CKOK	clock trim OK interrupt
CTC_INT_CKWARN	clock trim warning interrupt
CTC_INT_ERR	error interrupt
CTC_INT_EREF	expect reference interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable CTC clock trim OK interrupt */

ctc_interrupt_disable (CTC_INT_CKOK);

ctc_interrupt_flag_get

The description of ctc_interrupt_flag_get is shown as below:

Table 3-92. Function ctc_interrupt_flag_get

Function name	ctc_interrupt_flag_get
Function prototype	FlagStatus ctc_interrupt_flag_get(uint32_t interrupt);
Function descriptions	get CTC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	CTC interrupt flag

CTC_INT_FLAG_CKOK	clock trim OK interrupt
CTC_INT_FLAG_CKWARN	clock trim warning interrupt
CTC_INT_FLAG_ERR	error interrupt
CTC_INT_FLAG_EREF	expect reference interrupt
CTC_INT_FLAG_CKERR	clock trim error bit interrupt
CTC_INT_FLAG_REFMISS	reference sync pulse miss interrupt
CTC_INT_FLAG_TRIMERR	trim value error interrupt
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get CTC interrupt flag status */

FlagStatus state = ctc_interrupt_flag_get (CTC_INT_FLAG_CKOK);

ctc_interrupt_flag_clear

The description of ctc_interrupt_flag_clear is shown as below:

Table 3-93. Function ctc_interrupt_flag_clear

Function name	ctc_interrupt_flag_clear
Function prototype	void ctc_interrupt_flag_clear(uint32_t interrupt);
Function descriptions	clear CTC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	CTC interrupt flag
CTC_INT_FLAG_CKOK	clock trim OK interrupt
CTC_INT_FLAG_CKWARN	clock trim warning interrupt
CTC_INT_FLAG_ERR	error interrupt
CTC_INT_FLAG_EREF	expect reference interrupt
CTC_INT_FLAG_CKERR	clock trim error bit interrupt
CTC_INT_FLAG_REFMISS	reference sync pulse miss interrupt

CTC_INT_FLAG_TRIMERR	trim value error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*clear CTC interrupt flag status */

ctc_interrupt_flag_clear (CTC_INT_FLAG_CKOK);

ctc_flag_get

The description of ctc_flag_get is shown as below:

Table 3-94. Function ctc_flag_get

Function name	ctc_flag_get
Function prototype	FlagStatus ctc_flag_get(uint32_t flag);
Function descriptions	get CTC status flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	CTC status flag
CTC_FLAG_CKOK	clock trim OK interrupt flag
CTC_FLAG_CKWARN	clock trim warning interrupt flag
CTC_FLAG_ERR	error interrupt flag
CTC_FLAG_EREF	expect reference interrupt flag
CTC_FLAG_CKERR	clock trim error bit
CTC_FLAG_REFMISS	reference sync pulse miss flag
CTC_FLAG_TRIMERR	trim value error flag
Output parameter{out}	
-	**-**
Return value	
FlagStatus	SET or RESET

Example:

/* get CTC flag status */

FlagStatus state = ctc_flag_get (CTC_FLAG_CKOK);

ctc_flag_clear

The description of ctc_flag_clear is shown as below:

Table 3-95. Function ctc_flag_clear

Function name	ctc_flag_clear
Function prototype	void ctc_flag_clear (uint32_t flag);
Function descriptions	clear CTC status flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	CTC status flag
CTC_FLAG_CKOK	clock trim OK interrupt flag
CTC_FLAG_CKWARN	clock trim warning interrupt flag
CTC_FLAG_ERR	error interrupt flag
CTC_FLAG_EREF	expect reference interrupt flag
CTC_FLAG_CKERR	clock trim error bit
CTC_FLAG_REFMISS	reference sync pulse miss flag
CTC_FLAG_TRIMERR	trim value error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear CTC flag status */
ctc_flag_clear (CTC_FLAG_CKOK);
```

3.6. DAC

The Digital-to-analog converter converts 12-bit digital data to a voltage on the external pins. The DAC registers are listed in chapter *3.6.1*, the DAC firmware functions are introduced in chapter *3.6.2*.

3.6.1. Descriptions of Peripheral registers

DAC registers are listed in the table shown as below:

Table 3-96. DAC Registers

Registers	Descriptions
DAC_CTL0	DACx control register 0
DAC_SWT	DACx software trigger register
DAC_OUT0_R12DH	DACx_OUT0 12-bit right-aligned data holding register
DAC_OUT0_L12DH	DACx_OUT0 12-bit left-aligned data holding register
DAC_OUT0_R8DH	DACx_OUT0 8-bit right-aligned data holding register
DAC_OUT1_R12DH	DACx_OUT1 12-bit right-aligned data holding register

Registers	Descriptions
DAC_OUT1_L12DH	DACx_OUT1 12-bit left-aligned data holding register
DAC_OUT1_R8DH	DACx_OUT1 8-bit right-aligned data holding register
DACC_R12DH	DACx concurrent mode 12-bit right-aligned data holding register
DACC_L12DH	DACx concurrent mode 12-bit left-aligned data holding register
DACC_R8DH	DACx concurrent mode 8-bit right-aligned data holding register
DAC_OUT0_DO	DACx_OUT0 data output register
DAC_OUT1_DO	DACx_OUT1 data output register

3.6.2. Descriptions of Peripheral functions

DAC firmware functions are listed in the table shown as below:

Table 3-97. DAC firmware function

Function name	Function description
dac_deinit	deinitialize DAC
dac_enable	enable DAC
dac_disable	disable DAC
dac_dma_enable	enable DAC DMA function
dac_dma_disable	disable DAC DMA function
dac_output_buffer_enable	enable DAC output buffer
dac_output_buffer_disable	disable DAC output buffer
dac_output_value_get	get DAC output value
dac_data_set	set DAC data holding register value
dac_trigger_enable	enable DAC trigger
dac_trigger_disable	disable DAC trigger
dac_trigger_source_config	configure DAC trigger source
dac_software_trigger_enable	enable DAC software trigger
dac_wave_mode_config	configure DAC wave mode
dac_lfsr_noise_config	configure DAC LFSR noise mode
dac_triangle_noise_config	configure DAC triangle noise mode
dac_concurrent_enable	enable DAC concurrent mode
dac_concurrent_disable	disable DAC concurrent mode
dac_concurrent_software_trigger_enable	enable DAC concurrent software trigger
dac_concurrent_output_buffer_enable	enable DAC concurrent buffer function
dac_concurrent_output_buffer_disable	disable DAC concurrent buffer function
dac_concurrent_data_set	set DAC concurrent mode data holding register value

dac_deinit

The description of dac_deinit is shown as below:

Table 3-98. Function dac_deinit

Function name	dac_deinit

Function prototype	void dac_deinit(uint32_t dac_periph);
Function descriptions	deinitialize DAC
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize DAC0 */
dac_deinit(DAC0);
```

dac_enable

The description of dac_enable is shown as below:

Table 3-99. Function dac_enable

Function name	dac_enable
Function prototype	void dac_enable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	enable DAC
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC0_OUT0 */
dac_enable(DAC0, DAC_OUT0);
```

dac_disable

The description of dac_disable is shown as below:

Table 3-100. Function dac_disable

Function name	dac_disable
Function prototype	void dac_disable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	disable DAC
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC0_OUT0 */

dac_disable(DAC0, DAC_OUT0);
```

dac_dma_enable

The description of dac_dma_enable is shown as below:

Table 3-101. Function dac_dma_enable

Function name	dac_dma_enable
Function prototype	void dac_dma_enable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	enable DAC DMA function
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0,1)
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable DAC0_OUT0 DMA function */

dac_dma_enable(DAC0, DAC_OUT0);

dac_dma_disable

The description of dac_dma_disable is shown as below:

Table 3-102. Function dac_dma_disable

Function name	dac_dma_disable
Function prototype	void dac_dma_disable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	disable DAC DMA function
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DAC0_OUT0 DMA function */

dac_dma_disable(DAC0, DAC_OUT0);

dac_output_buffer_enable

The description of dac_output_buffer_enable is shown as below:

Table 3-103. Function dac_output_buffer_enable

Function name	dac_output_buffer_enable
Function prototype	void dac_output_buffer_enable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	enable DAC output buffer
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral

DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC0_OUT0 output buffer */
dac_output_buffer_enable(DAC0, DAC_OUT0);
```

dac_output_buffer_disable

The description of dac_output_buffer_disable is shown as below:

Table 3-104. Function dac_output_buffer_disable

Function name	dac_output_buffer_disable
Function prototype	void dac_output_buffer_disable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	disable DAC output buffer
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC0_OUT0 output buffer */
dac_output_buffer_disable(DAC0, DAC_OUT0);
```

dac_output_value_get

The description of dac_output_value_get is shown as below:

Table 3-105. Function dac_output_value_get

Function name	dac_output_value_get
Function prototype	uint16_t dac_output_value_get(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	get DAC output value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	
uint16_t	DAC output data (0~4095)

Example:

```
/* get the DAC0_OUT0 last data output value */
uint16_t data=0;
data = dac_output_value_get(DAC0, DAC_OUT0);
```

dac_data_set

The description of dac_data_set is shown as below:

Table 3-106. Function dac_data_set

Function name	dac_data_set
Function prototype	void dac_data_set(uint32_t dac_periph, uint8_t dac_out, uint32_t dac_align, uint16_t data);
Function descriptions	set DAC data holding register value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0,1)
Input parameter{in}	
dac_align	DAC data alignment mode
DAC_ALIGN_12B_R	12-bit right-aligned data

DAC_ALIGN_12B_L	12-bit left-aligned data
DAC_ALIGN_8B_R	8-bit right-aligned data
Input parameter{in}	
data	data to be loaded (0~4095)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC0_OUT0 data holding register value */
dac_data_set(DAC0, DAC_OUT0, DAC_ALIGN_8B_R, 0xFF);
```

dac_trigger_enable

The description of dac_trigger_enable is shown as below:

Table 3-107. Function dac_trigger_enable

Function name	dac_trigger_enable
Function prototype	void dac_trigger_enable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	enable DAC trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC0_OUT0 trigger */
dac_trigger_enable(DAC0, DAC_OUT0);
```

dac_trigger_disable

The description of dac_trigger_disable is shown as below:

Table 3-108. Function dac_trigger_disable

Function name	dac_trigger_disable

Function prototype	void dac_trigger_disable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	disable DAC trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DAC0_OUT0 trigger */

dac_trigger_disable(DAC0, DAC_OUT0);

dac_trigger_source_config

The description of dac_trigger_source_config is shown as below:

Table 3-109. Function dac_trigger_source_config

Function name	dac_trigger_source_config
Function prototype	void dac_trigger_source_config(uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);
Function descriptions	configure DAC trigger source
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0,1)
Input parameter{in}	
triggersource	external trigger of DAC
DAC_TRIGGER_T5_TRGO	TIMER5 TRGO
DAC_TRIGGER_T2_TRGO	TIMER2 TRGO
DAC_TRIGGER_T6_TRG	TIMER6 TRGO

O	
DAC_TRIGGER_T4_TRGO	TIMER4 TRGO
DAC_TRIGGER_T1_TRGO	TIMER1 TRGO
DAC_TRIGGER_T3_TRGO	TIMER3 TRGO
DAC_TRIGGER_EXTI_9	EXTI interrupt line9 event
DAC_TRIGGER_SOFTWARE	software trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC0_OUT0 trigger source */
dac_trigger_source_config(DAC0, DAC_OUT0, DAC_TRIGGER_T1_TRGO);
```

dac_software_trigger_enable

The description of dac_software_trigger_enable is shown as below:

Table 3-110. Function dac_software_trigger_enable

Function name	dac_software_trigger_enable
Function prototype	void dac_software_trigger_enable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	enable DAC software trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC0_OUT0 software trigger */
```

dac_software_trigger_enable(DAC0, DAC_OUT0);

dac_wave_mode_config

The description of dac_wave_mode_config is shown as below:

Table 3-111. Function dac_wave_mode_config

Function name	dac_wave_mode_config
Function prototype	void dac_wave_mode_config(uint32_t dac_periph, uint8_t dac_out, uint32_t wave_mode);
Function descriptions	configure DAC wave mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0,1)
Input parameter{in}	
wave_mode	DAC wave mode
DAC_WAVE_DISABLE	wave mode disable
DAC_WAVE_MODE_LFSR	LFSR noise mode
DAC_WAVE_MODE_TRIANGLE	triangle noise mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure DAC0_OUT0 wave mode */

dac_wave_mode_config(DAC0, DAC_OUT0, DAC_WAVE_DISABLE);

dac_lfsr_noise_config

The description of dac_lfsr_noise_config is shown as below:

Table 3-112. Function dac_lfsr_noise_config

Function name	dac_lfsr_noise_config
Function prototype	void dac_lfsr_noise_config(uint32_t dac_periph, uint8_t dac_out, uint32_t unmask_bits);
Function descriptions	configure DAC LFSR noise mode
Precondition	-

The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0,1)
Input parameter{in}	
unmask_bits	LFSR noise unmask bits
DAC_LFSR_BIT0	unmask the LFSR bit0
DAC_LFSR_BITSx_0	unmask the LFSR bits [x:0] (x = 1,2,3..11)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC0_OUT0 LFSR noise mode */
dac_lfsr_noise_config(DAC0, DAC_OUT0, DAC_LFSR_BIT0);
```

dac_triangle_noise_config

The description of dac_triangle_noise_config is shown as below:

Table 3-113. Function dac_triangle_noise_config

Function name	dac_triangle_noise_config
Function prototype	void dac_triangle_noise_config(uint32_t dac_periph, uint8_t dac_out, uint32_t amplitude);
Function descriptions	configure DAC triangle noise mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Input parameter{in}	
dac_out	DAC output
DAC_OUTx	DAC output channel selection (x = 0,1)
Input parameter{in}	
amplitude	the amplitude of the triangle
DAC_TRIANGLE_AMPLITUDE_x	x = 2^n-1 (n = 1..12)
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* configure DAC0_OUT0 triangle noise mode */

dac_triangle_noise_config(DAC0, DAC_OUT0, DAC_TRIANGLE_AMPLITUDE_1);

dac_concurrent_enable

The description of dac_concurrent_enable is shown as below:

Table 3-114. Function dac_concurrent_enable

Function name	dac_concurrent_enable
Function prototype	void dac_concurrent_enable(uint32_t dac_periph);
Function descriptions	enable DAC concurrent mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC0 concurrent mode */

dac_concurrent_enable(DAC0);

dac_concurrent_disable

The description of dac_concurrent_disable is shown as below:

Table 3-115. Function dac_concurrent_disable

Function name	dac_concurrent_disable
Function prototype	void dac_concurrent_disable(uint32_t dac_periph);
Function descriptions	disable DAC concurrent mode
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* disable DAC0 concurrent mode */

dac_concurrent_disable(DAC0);

dac_concurrent_software_trigger_enable

The description of dac_concurrent_software_trigger_enable is shown as below:

Table 3-116. Function dac_concurrent_software_trigger_enable

Function name	dac_concurrent_software_trigger_enable
Function prototype	void dac_concurrent_software_trigger_enable(uint32_t dac_periph);
Function descriptions	enable DAC concurrent software trigger
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable DAC0 concurrent software trigger */

dac_concurrent_software_trigger_enable(DAC0);

dac_concurrent_output_buffer_enable

The description of dac_concurrent_output_buffer_enable is shown as below:

Table 3-117. Function dac_concurrent_output_buffer_enable

Function name	dac_concurrent_output_buffer_enable
Function prototype	void dac_concurrent_output_buffer_enable(uint32_t dac_periph);
Function descriptions	enable DAC concurrent buffer function
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* enable DAC0 concurrent buffer function */

dac_concurrent_output_buffer_enable(DAC0);

dac_concurrent_output_buffer_disable

The description of dac_concurrent_output_buffer_disable is shown as below:

Table 3-118. Function dac_concurrent_output_buffer_disable

Function name	dac_concurrent_output_buffer_disable
Function prototype	void dac_concurrent_output_buffer_disable(uint32_t dac_periph);
Function descriptions	disable DAC concurrent buffer function
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection (x = 0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable DAC0 concurrent buffer function */

dac_concurrent_output_buffer_disable(DAC0);

dac_concurrent_data_set

The description of dac_concurrent_data_set is shown as below:

Table 3-119. Function dac_concurrent_data_set

Function name	dac_concurrent_data_set
Function prototype	void dac_concurrent_data_set(uint32_t dac_periph, uint32_t dac_align, uint16_t data0, uint16_t data1);
Function descriptions	set DAC concurrent mode data holding register value
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0)

Input parameter{in}	
dac_align	DAC data alignment mode
DAC_ALIGN_8B_R	8-bit right-aligned data
DAC_ALIGN_12B_R	12-bit right-aligned data
DAC_ALIGN_12B_L	12-bit left-aligned data
Input parameter{in}	
data0	DACx_OUT0 data to be loaded (0~4095)
Input parameter{in}	
data1	DACx_OUT1 data to be loaded (0~4095)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC0 concurrent mode data holding register value */
dac_concurrent_data_set(DAC0, DAC_ALIGN_8B_R, 0xFF, 0xFF);
```

3.7. DBG

The DBG hold unit helps debugger to debug power saving mode. The DBG registers are listed in chapter *3.7.1*. the DBG firmware functions are introduced in chapter *3.7.2*.

3.7.1. Descriptions of Peripheral registers

DBG registers are listed in the table shown as below:

Table 3-120. DBG Registers

Registers	Descriptions
DBG_ID	DBG ID code register
DBG_CTL	DBG control register

3.7.2. Descriptions of Peripheral functions

DBG firmware functions are listed in the table shown as below:

Table 3-121. DBG firmware function

Function name	Function description
dbg_id_get	read DBG_ID code register
dbg_low_power_enable	enable low power behavior when the MCU is in debug mode
dbg_low_power_disable	disable low power behavior when the MCU is in debug mode

Function name	Function description
dbg_periph_enable	enable peripheral behavior when the MCU is in debug mode
dbg_periph_disable	disable peripheral behavior when the MCU is in debug mode
dbg_trace_pin_enable	enable trace pin assignment
dbg_trace_pin_disable	disable trace pin assignment

Enum dbg_periph_enum

Table 3-122. Enum dbg_periph_enum

Member name	Function description
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_TIMER0_HOLD	hold TIMER0 counter when core is halted
DBG_TIMER1_HOLD	hold TIMER1 counter when core is halted
DBG_TIMER2_HOLD	hold TIMER2 counter when core is halted
DBG_TIMER3_HOLD	hold TIMER3 counter when core is halted
DBG_I2C0_HOLD	hold I2C0 smbus when core is halted
DBG_I2C1_HOLD	hold I2C1 smbus when core is halted
DBG_TIMER4_HOLD	hold TIMER4 counter when core is halted
DBG_TIMER5_HOLD	hold TIMER5 counter when core is halted
DBG_TIMER6_HOLD	hold TIMER6 counter when core is halted
DBG_TIMER7_HOLD	hold TIMER7 counter when core is halted
DBG_TIMER11_HOLD	hold TIMER11 counter when core is halted
DBG_TIMER12_HOLD	hold TIMER12 counter when core is halted
DBG_TIMER13_HOLD	hold TIMER13 counter when core is halted
DBG_TIMER8_HOLD	hold TIMER8 counter when core is halted
DBG_TIMER9_HOLD	hold TIMER9 counter when core is halted
DBG_TIMER10_HOLD	hold TIMER10 counter when core is halted

dbg_id_get

The description of dbg_id_get is shown as below:

Table 3-123. Function dbg_id_get

Function name	dbg_id_get
Function prototype	uint32_t dbg_id_get(void);
Function descriptions	Read DBG_ID code register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

uint32_t	DBG_ID code (0-0xFFFFFFFF)

Example:

```
/* read DBG_ID code register */
uint32_t    id_value = 0;
id_value = dbg_id_get();
```

dbg_low_power_enable

The description of dbg_low_power_enable is shown as below:

Table 3-124. Function dbg_low_power_enable

Function name	dbg_low_power_enable
Function prototype	void dbg_low_power_enable(uint32_t dbg_low_power);
Function descriptions	Enable low power behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_low_power	low power mode
DBG_LOW_POWER_SLEEP	keep debugger connection during sleep mode
DBG_LOW_POWER_DEEPSLEEP	keep debugger connection during deepsleep mode
DBG_LOW_POWER_STANDBY	keep debugger connection during standby mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable low power behavior when the mcu is in debug mode */
dbg_low_power_enable(DBG_LOW_POWER_SLEEP);
```

dbg_low_power_disable

The description of dbg_low_power_disable is shown as below:

Table 3-125. Function dbg_low_power_disable

Function name	dbg_low_power_disable
Function prototype	void dbg_low_power_disable(uint32_t dbg_low_power);
Function descriptions	Disable low power behavior when the mcu is in debug mode
Precondition	-

The called functions	-
Input parameter{in}	
dbg_low_power	low power mode
DBG_LOW_POWER_SLEEP	keep debugger connection during sleep mode
DBG_LOW_POWER_DEEPSLEEP	keep debugger connection during deepsleep mode
DBG_LOW_POWER_STANDBY	keep debugger connection during standby mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable low power behavior when the mcu is in debug mode */
dbg_low_power_disable(DBG_LOW_POWER_SLEEP);
```

dbg_periph_enable

The description of dbg_periph_enable is shown as below:

Table 3-126. Function dbg_periph_enable

Function name	dbg_periph_enable
Function prototype	void dbg_periph_enable(dbg_periph_enum dbg_periph);
Function descriptions	Enable peripheral behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_periph	Peripheral refer to ***Table 3-122. Enum dbg_periph_enum***
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_I2Cx_HOLD	x=0,1, hold I2Cx smbus when core is halted
DBG_TIMERx_HOLD	x=0,1,2,3,4,5,6,7,8,9,10,11,12,13, hold TIMERx counter when core is halted
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable peripheral behavior when the mcu is in debug mode */
dbg_periph_enable(DBG_TIMER0_HOLD);
```

dbg_periph_disable

The description of dbg_periph_disable is shown as below:

Table 3-127. Function dbg_periph_disable

Function name	dbg_periph_disable
Function prototype	void dbg_periph_disable(dbg_periph_enum dbg_periph);
Function descriptions	Disable peripheral behavior when the mcu is in debug mode
Precondition	-
The called functions	-
Input parameter{in}	
dbg_periph	peripheral refer to *Table 3-122. Enum dbg_periph_enum*
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_I2Cx_HOLD	x=0,1, hold I2Cx smbus when core is halted
DBG_TIMERx_HOLD	x=0,1,2,3,4,5,6,7,8,9,10,11,12,13, hold TIMERx counter when core is halted
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable peripheral behavior when the mcu is in debug mode */
dbg_periph_disable(DBG_TIMER0_HOLD);
```

dbg_trace_pin_enable

The description of dbg_trace_pin_enable is shown as below:

Table 3-128. Function dbg_trace_pin_enable

Function name	dbg_trace_pin_enable
Function prototype	void dbg_trace_pin_enable(void);
Function descriptions	Enable trace pin assignment
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable trace pin assignment */
```

dbg_trace_pin_enable();

dbg_trace_pin_disable

The description of dbg_trace_pin_disable is shown as below:

Table 3-129. Function dbg_trace_pin_disable

Function name	dbg_trace_pin_disable
Function prototype	void dbg_trace_pin_disable(void);
Function descriptions	Disable trace pin assignment
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable trace pin assignment */
dbg_trace_pin_disable();
```

3.8. DMA

The direct memory access (DMA) controller provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. The DMA registers are listed in chapter *3.8.1*, the DMA firmware functions are introduced in chapter *3.8.2*.

3.8.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

Table 3-130. DMA Registers

Registers	Descriptions
DMA_INTF	Interrupt flag register
DMA_INTC	Interrupt flag clear register
DMA_CHxCTL (x=0..6)	Channel x control register
DMA_CHxCNT (x=0..6)	Channel x counter register

Registers	Descriptions
DMA_CHxPADDR (x=0..6)	Channel x peripheral base address register
DMA_CHxMADDR (x=0..6)	Channel x memory base address register

3.8.2. Descriptions of Peripheral functions

DMA firmware functions are listed in the table shown as below:

Table 3-131. DMA firmware function

Function name	Function description
dma_deinit	deinitialize DMA a channel registers
dma_struct_para_init	initialize the parameters of DMA struct with the default values
dma_init	initialize DMA channel
dma_circulation_enable	enable DMA circulation mode
dma_circulation_disable	disable DMA circulation mode
dma_memory_to_memory_enable	enable memory to memory mode
dma_memory_to_memory_disable	disable memory to memory mode
dma_channel_enable	enable DMA channel
dma_channel_disable	disable DMA channel
dma_periph_address_config	set DMA peripheral base address
dma_memory_address_config	set DMA memory base address
dma_transfer_number_config	set the number of remaining data to be transferred by the DMA
dma_transfer_number_get	get the number of remaining data to be transferred by the DMA
dma_priority_config	configure priority level of DMA channel
dma_memory_width_config	configure transfer data size of memory
dma_periph_width_config	configure transfer data size of peripheral
dma_memory_increase_enable	enable next address increasement algorithm of memory
dma_memory_increase_disable	disable next address increasement algorithm of memory
dma_periph_increase_enable	enable next address increasement algorithm of peripheral
dma_periph_increase_disable	disable next address increasement algorithm of peripheral
dma_transfer_direction_config	configure the direction of data transfer on the channel
dma_flag_get	check DMA flag is set or not
dma_flag_clear	clear the flag of a DMA channel
dma_interrupt_flag_get	check DMA flag and interrupt enable bit is set or not
dma_interrupt_flag_clear	clear the interrupt flag of a DMA channel
dma_interrupt_enable	enable DMA interrupt
dma_interrupt_disable	disable DMA interrupt

Structure dma_parameter_struct

Table 3-132. Structure dma_parameter_struct

Member name	Function description
periph_addr	peripheral base address
periph_width	transfer data size of peripheral
memory_addr	memory base address
memory_width	transfer data size of memory
number	channel transfer number
priority	channel priority level
periph_inc	peripheral increasing mode
memory_inc	memory increasing mode
direction	channel data transfer direction

dma_deinit

The description of dma_deinit is shown as below:

Table 3-133. Function dma_deinit

Function name	dma_deinit
Function prototype	void dma_deinit(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	deinitialize DMA a channel registers
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 channel0 initialize */
dma_deinit(DMA0, DMA_CH0);
```

dma_struct_para_init

The description of dma_struct_para_init is shown as below:

Table 3-134. Function dma_struct_para_init

Function name	dma_struct_para_init
Function prototype	void dma_struct_para_init(dma_parameter_struct* init_struct);
Function descriptions	initialize the parameters of DMA struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
***init_struct**	a spi_parameter_struct address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of DMA */
dma_parameter_struct dma_init_struct;
dma_struct_para_init(&dma_init_struct);
```

dma_init

The description of dma_init is shown as below:

Table 3-135. Function dma_init

Function name	dma_init
Function prototype	void dma_init(uint32_t dma_periph, dma_channel_enum channelx, dma_parameter_struct* init_struct);
Function descriptions	initialize DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to ***Table 3-132. Structure dma_parameter_struct***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 channel0 initialize */
dma_parameter_struct dma_init_struct;
dma_deinit(DMA0, DMA_CH0);
dma_struct_para_init(&dma_init_struct);

dma_init_struct.direction = DMA_PERIPHERAL_TO_MEMORY;
dma_init_struct.memory_addr = (uint32_t)g_destbuf;
dma_init_struct.memory_inc = DMA_MEMORY_INCREASE_ENABLE;
dma_init_struct.memory_width = DMA_MEMORY_WIDTH_8BIT;
dma_init_struct.number = TRANSFER_NUM;
dma_init_struct.periph_addr = (uint32_t)BANK0_WRITE_START_ADDR;
dma_init_struct.periph_inc = DMA_PERIPH_INCREASE_ENABLE;
dma_init_struct.periph_width = DMA_PERIPHERAL_WIDTH_8BIT;
dma_init_struct.priority = DMA_PRIORITY_ULTRA_HIGH;
dma_init(DMA0, DMA_CH0, &dma_init_struct);
```

dma_circulation_enable

The description of dma_circulation_enable is shown as below:

Table 3-136. Function dma_circulation_enable

Function name	dma_circulation_enable
Function prototype	void dma_circulation_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA circulation mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 channel0 circulation mode */
dma_circulation_enable(DMA0, DMA_CH0);
```

dma_circulation_disable

The description of dma_circulation_disable is shown as below:

Table 3-137. Function dma_circulation_disable

Function name	dma_circulation_disable
Function prototype	void dma_circulation_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable DMA circulation mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA0 channel0 circulation mode */
dma_circulation_disable(DMA0, DMA_CH0);
```

dma_memory_to_memory_enable

The description of dma_memory_to_memory_enable is shown as below:

Table 3-138. Function dma_memory_to_memory_enable

Function name	dma_memory_to_memory_enable
Function prototype	void dma_memory_to_memory_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable memory to memory mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..	DMA channel selection

6; DMA1: x=0..4)	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 channel0 memory to memory mode */
dma_memory_to_memory_enable(DMA0, DMA_CH0);
```

dma_memory_to_memory_disable

The description of dma_memory_to_memory_disable is shown as below:

Table 3-139. Function dma_memory_to_memory_disable

Function name	dma_memory_to_memory_disable
Function prototype	void dma_memory_to_memory_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable memory to memory mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA0 channel0 memory to memory mode */
dma_memory_to_memory_enable(DMA0, DMA_CH0);
```

dma_channel_enable

The description of dma_channel_enable is shown as below:

Table 3-140. Function dma_channel_enable

Function name	dma_channel_enable
Function prototype	void dma_channel_enable(uint32_t dma_periph, dma_channel_enum

	channelx);
Function descriptions	enable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 channel0 */
dma_channel_enable(DMA0, DMA_CH0);
```

dma_channel_disable

The description of dma_channel_disable is shown as below:

Table 3-141. Function dma_channel_disable

Function name	dma_channel_disable
Function prototype	void dma_channel_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA0 channel0 */
dma_channel_disable(DMA0, DMA_CH0);
```

dma_periph_address_config

The description of dma_periph_address_config is shown as below:

Table 3-142. Function dma_periph_address_config

Function name	dma_periph_address_config
Function prototype	void dma_periph_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA peripheral base address
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
address	peripheral base address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
#define BANK0_WRITE_START_ADDR                ((uint32_t)0x08004000)

dma_periph_address_config(DMA0, DMA_CH0, BANK0_WRITE_START_ADDR);
```

dma_memory_address_config

The description of dma_memory_address_config is shown as below:

Table 3-143. Function dma_memory_address_config

Function name	dma_memory_address_config
Function prototype	void dma_memory_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA memory base address
Precondition	-
The called functions	-
Input parameter{in}	

dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
address	memory base address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
uint8_t g_destbuf[TRANSFER_NUM];

dma_memory_address_config(DMA0, DMA_CH0, (uint32_t) g_destbuf);
```

dma_transfer_number_config

The description of dma_transfer_number_config is shown as below:

Table 3-144. Function dma_transfer_number_config

Function name	dma_transfer_number_config
Function prototype	void dma_transfer_number_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t number);
Function descriptions	set the number of remaining data to be transferred by the DMA
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
number	data transfer number
0-0xffff	number
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
#define TRANSFER_NUM                    0x400

dma_transfer_number_config(DMA0, DMA_CH0, TRANSFER_NUM);
```

dma_transfer_number_get

The description of dma_transfer_number_get is shown as below:

Table 3-145. Function dma_transfer_number_get

Function name	dma_transfer_number_get
Function prototype	uint32_t dma_transfer_number_get(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	get the number of remaining data to be transferred by the DMA
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
uint32_t	0-0xffff

Example:

```
uint32_t number = 0;

number = dma_transfer_number_get(DMA0, DMA_CH0);
```

dma_priority_config

The description of dma_priority_config is shown as below:

Table 3-146. Function dma_priority_config

Function name	dma_priority_config
Function prototype	void dma_priority_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t priority);
Function descriptions	configure priority level of DMA channel
Precondition	-
The called functions	-
Input parameter{in}	

dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
priority	priority Level of this channel
DMA_PRIORITY_LOW	low priority
DMA_PRIORITY_MEDIUM	medium priority
DMA_PRIORITY_HIGH	high priority
DMA_PRIORITY_ULTRA_HIGH	ultra high priority
Output parameter{out}	
-	-
Return value	
-	-

Example:

dma_priority_config(DMA0, DMA_CH0, DMA_PRIORITY_ULTRA_HIGH);

dma_memory_width_config

The description of dma_memory_width_config is shown as below:

Table 3-147. Function dma_memory_width_config

Function name	dma_memory_width_config
Function prototype	void dma_memory_width_config (uint32_t dma_periph, dma_channel_enum channelx, uint32_t mwidth);
Function descriptions	configure transfer data size of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
mwidth	transfer data width of memory
DMA_MEMORY_WIDT	transfer data width of memory is 8-bit

H_8BIT	
DMA_MEMORY_WIDTH_16BIT	transfer data width of memory is 16-bit
DMA_MEMORY_WIDTH_32BIT	transfer data width of memory is 32-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
dma_memory_width_config(DMA0, DMA_CH0, DMA_MEMORY_WIDTH_8BIT);
```

dma_periph_width_config

The description of dma_periph_width_config is shown as below:

Table 3-148. Function dma_periph_width_config

Function name	dma_periph_width_config
Function prototype	void dma_periph_width_config (uint32_t dma_periph, dma_channel_enum channelx, uint32_t pwidth);
Function descriptions	configure transfer data size of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
pwidth	transfer data width of peripheral
DMA_PERIPHERAL_WIDTH_8BIT	transfer data width of peripheral is 8-bit
DMA_PERIPHERAL_WIDTH_16BIT	transfer data width of peripheral is 16-bit
DMA_PERIPHERAL_WIDTH_32BIT	transfer data width of peripheral is 32-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
dma_periph_width_config(DMA0, DMA_CH0, DMA_PERIPHERAL_WIDTH_8BIT);
```

dma_memory_increase_enable

The description of dma_memory_increase_enable is shown as below:

Table 3-149. Function dma_memory_increase_enable

Function name	dma_memory_increase_enable
Function prototype	void dma_memory_increase_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
dma_memory_increase_enable(DMA0, DMA_CH0);
```

dma_memory_increase_disable

The description of dma_memory_increase_disable is shown as below:

Table 3-150. Function dma_memory_increase_disable

Function name	dma_memory_increase_disable
Function prototype	void dma_memory_increase_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	

channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
dma_memory_increase_disable(DMA0, DMA_CH0);
```

dma_periph_increase_enable

The description of dma_periph_increase_enable is shown as below:

Table 3-151. Function dma_periph_increase_enable

Function name	dma_periph_increase_enable
Function prototype	void dma_periph_increase_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
dma_periph_increase_enable(DMA0, DMA_CH0);
```

dma_periph_increase_disable

The description of dma_periph_increase_disable is shown as below:

Table 3-152. Function dma_periph_increase_disable

Function name	dma_periph_increase_disable
Function prototype	void dma_periph_increase_disable(uint32_t dma_periph,

	dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

dma_periph_increase_disable(DMA0, DMA_CH0);

dma_transfer_direction_config

The description of dma_transfer_direction_config is shown as below:

Table 3-153. Function dma_transfer_direction_config

Function name	dma_transfer_direction_config
Function prototype	void dma_transfer_direction_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t direction);
Function descriptions	configure the direction of data transfer on the channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
direction	specify the direction of data transfer
DMA_PERIPHERAL_TO_MEMORY	read from peripheral and write to memory
DMA_MEMORY_TO_PERIPHERAL	read from memory and write to peripheral

Output parameter{out}	
-	-
Return value	
-	-

Example:

dma_transfer_direction_config(DMA0, DMA_CH0, DMA_PERIPHERAL_TO_MEMORY);

dma_flag_get

The description of dma_flag_get is shown as below:

Table 3-154. Function dma_flag_get

Function name	dma_flag_get
Function prototype	FlagStatus dma_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
flag	specify get which flag
DMA_FLAG_G	global interrupt flag of channel
DMA_FLAG_FTF	full transfer finish flag of channel
DMA_FLAG_HTF	half transfer finish flag of channel
DMA_FLAG_ERR	error flag of channel
Output parameter{out}	
-	**-**
Return value	
FlagStatus	SET or RESET

Example:

FlagStatus flag = RESET;

flag = dma_flag_get(DMA0, DMA_CH0, DMA_FLAG_FTF);

dma_flag_clear

The description of dma_flag_clear is shown as below:

Table 3-155. Function dma_flag_clear

Function name	dma_flag_clear
Function prototype	void dma_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear the flag of a DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
flag	specify get which flag
DMA_FLAG_G	global interrupt flag of channel
DMA_FLAG_FTF	full transfer finish flag of channel
DMA_FLAG_HTF	half transfer finish flag of channel
DMA_FLAG_ERR	error flag of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
dma_flag_clear(DMA0, DMA_CH0, DMA_FLAG_FTF);
```

dma_interrupt_flag_get

The description of dma_interrupt_flag_get is shown as below:

Table 3-156. Function dma_interrupt_flag_get

Function name	dma_interrupt_flag_get
Function prototype	FlagStatus dma_interrupt_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag and interrupt enable bit is set or not
Precondition	-
The called functions	-
Input parameter{in}	

dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
flag	specify get which flag
DMA_INT_FLAG_FTF	full transfer finish interrupt flag of channel
DMA_INT_FLAG_HTF	half transfer finish interrupt flag of channel
DMA_INT_FLAG_ERR	error interrupt flag of channel
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
if(dma_interrupt_flag_get(DMA0, DMA_CH3, DMA_INT_FLAG_FTF)){
    dma_interrupt_flag_clear(DMA0, DMA_CH3, DMA_INT_FLAG_G);
}
```

dma_interrupt_flag_clear

The description of dma_interrupt_flag_clear is shown as below:

Table 3-157. Function dma_interrupt_flag_clear

Function name	dma_interrupt_flag_clear
Function prototype	void dma_interrupt_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear the interrupt flag of a DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
flag	specify get which flag
DMA_INT_FLAG_G	global interrupt flag of channel
DMA_INT_FLAG_FTF	full transfer finish interrupt flag of channel

DMA_INT_FLAG_HTF	half transfer finish interrupt flag of channel
DMA_INT_FLAG_ERR	error interrupt flag of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
if(dma_interrupt_flag_get(DMA0, DMA_CH3, DMA_INT_FLAG_FTF)){
    dma_interrupt_flag_clear(DMA0, DMA_CH3, DMA_INT_FLAG_G);
}
```

dma_interrupt_enable

The description of dma_interrupt_enable is shown as below:

Table 3-158. Function dma_interrupt_enable

Function name	dma_interrupt_enable
Function prototype	void dma_interrupt_enable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t source);
Function descriptions	enable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
source	DMA interrupt source
DMA_INT_FTF	full transfer finish interrupt of channel
DMA_INT_HTF	half transfer finish interrupt of channel
DMA_INT_ERR	error interrupt of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 channel0 interrupt configuration */
dma_interrupt_enable(DMA0, DMA_CH0, DMA_INT_FTF);
```

dma_interrupt_disable

The description of dma_interrupt_disable is shown as below:

Table 3-159. Function dma_interrupt_disable

Function name	dma_interrupt_disable
Function prototype	void dma_interrupt_disable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t source);
Function descriptions	disable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(DMA0:x=0..6; DMA1: x=0..4)	DMA channel selection
Input parameter{in}	
source	DMA interrupt source
DMA_INT_FTF	full transfer finish interrupt of channel
DMA_INT_HTF	half transfer finish interrupt of channel
DMA_INT_ERR	error interrupt of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 channel0 interrupt configuration */
dma_interrupt_disable(DMA0, DMA_CH0, DMA_INT_FTF);
```

3.9. EXMC

The external memory controller EXMC, is used as a translator for MCU to access a variety of external memory. The EXMC registers are listed in chapter *3.9.1*, the EXMC firmware functions are introduced in chapter *3.9.2*.

3.9.1. Descriptions of Peripheral registers

EXMC registers are listed in the table shown as below:

Table 3-160. EXMC Registers

Registers	Descriptions
EXMC_SNCTL	SRAM/NOR Flash control registers
EXMC_SNTCFG	SRAM/NOR Flash timing configuration registers
EXMC_SNWTCFG	SRAM/NOR Flash write timing configuration registers

3.9.2. Descriptions of Peripheral functions

EXMC firmware functions are listed in the table shown as below:

Table 3-161. EXMC firmware function

Function name	Function description
exmc_norsram_deinit	deinitialize EXMC NOR/SRAM bank
exmc_norsram_init	initialize EXMC NOR/SRAM bank
exmc_norsram_struct_para_init	initialize the struct exmc_norsram_parameter_struct
exmc_norsram_enable	enable EXMC NOR/PSRAM bank
exmc_norsram_disable	disable EXMC NOR/PSRAM bank
exmc_norsram_page_size_config	configure CRAM page size

Structure exmc_norsram_timing_parameter_struct

Table 3-162. Structure exmc_norsram_timing_parameter_struct

Member name	Function description
asyn_access_mode	asynchronous access mode
syn_data_latency	configure the data latency
syn_clk_division	configure the clock divide ratio
bus_latency	configure the bus latency
asyn_data_setuptime	configure the data setup time,asynchronous access mode valid
asyn_address_holdtime	configure the address hold time,asynchronous access mode valid
asyn_address_setuptime	configure the data setup time,asynchronous access mode valid

Structure exmc_norsram_parameter_struct

Table 3-163. Structure exmc_norsram_parameter_struct

Member name	Function description
write_mode	the write mode, synchronous mode or asynchronous mode
extended_mode	enable or disable the extended mode
asyn_wait	enable or disable the asynchronous wait function
nwait_signal	enable or disable the NWAIT signal while in synchronous bust mode
memory_write	enable or disable the write operation
nwait_config	NWAIT signal configuration

wrap_burst_mode	enable or disable the wrap burst mode
nwait_polarity	specifies the polarity of NWAIT signal from memory
burst_mode	enable or disable the burst mode
databus_width	specifies the databus width of external memory
memory_type	specifies the type of external memory
address_data_mux	specifies whether the data bus and address bus are multiplexed
read_write_timing	timing parameters for read and write if the extended mode is not used or the timing parameters for read if the extended mode is used
write_timing	timing parameters for write when the extended mode is used

exmc_norsram_deinit

The description of exmc_norsram_deinit is shown as below:

Table 3-164. Function exmc_norsram_deinit

Function name	exmc_norsram_deinit
Function prototype	void exmc_norsram_deinit(void);
Function descriptions	deinitialize EXMC NOR/SRAM bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize EXMC NOR/SRAM bank */

exmc_norsram_deinit();
```

exmc_norsram_init

The description of exmc_norsram_init is shown as below:

Table 3-165. Function exmc_norsram_init

Function name	exmc_norsram_init
Function prototype	void exmc_norsram_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
Function descriptions	initialize EXMC NOR/SRAM bank
Precondition	-
The called functions	-
Input parameter{in}	
exmc_norsram_init_st	Structure for initialization, the structure members can refer to ***Table 3-163.***

ruct	*Structure exmc_norsram_parameter_struct*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize EXMC NOR/SRAM bank */
exmc_norsram_parameter_struct lcd_init_struct;
exmc_norsram_timing_parameter_struct lcd_timing_init_struct;
/* configure timing parameter */
lcd_timing_init_struct.asyn_access_mode = EXMC_ACCESS_MODE_A;
lcd_timing_init_struct.syn_data_latency = EXMC_DATALAT_2_CLK;
lcd_timing_init_struct.syn_clk_division = EXMC_SYN_CLOCK_RATIO_DISABLE;
lcd_timing_init_struct.bus_latency = 1;
lcd_timing_init_struct.asyn_data_setuptime = 5;
lcd_timing_init_struct.asyn_address_holdtime = 2;
lcd_timing_init_struct.asyn_address_setuptime = 2;
/* configure EXMC bus parameters */
lcd_init_struct.write_mode = EXMC_ASYN_WRITE;
lcd_init_struct.extended_mode = DISABLE;
lcd_init_struct.asyn_wait = DISABLE;
lcd_init_struct.nwait_signal = DISABLE;
lcd_init_struct.memory_write = ENABLE;
lcd_init_struct.nwait_config = EXMC_NWAIT_CONFIG_BEFORE;
lcd_init_struct.wrap_burst_mode = DISABLE;
lcd_init_struct.nwait_polarity = EXMC_NWAIT_POLARITY_LOW;
lcd_init_struct.burst_mode = DISABLE;
lcd_init_struct.databus_width = EXMC_NOR_DATABUS_WIDTH_16B;
lcd_init_struct.memory_type = EXMC_MEMORY_TYPE_SRAM;
lcd_init_struct.address_data_mux = DISABLE;
```

lcd_init_struct.read_write_timing = &lcd_timing_init_struct;

lcd_init_struct.write_timing = &lcd_timing_init_struct;

exmc_norsram_init(&lcd_init_struct);

exmc_norsram_struct_para_init

The description of exmc_norsram_struct_para_init is shown as below:

Table 3-166. Function exmc_norsram_struct_para_init

Function name	exmc_norsram_struct_para_init
Function prototype	void exmc_norsram_struct_para_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
Function descriptions	initialize the struct exmc_norsram_parameter_struct
Precondition	-
The called functions	-
Input parameter{in}	
exmc_norsram_init_struct	Structure for initialization, the structure members can refer to *Table 3-163. Structure exmc_norsram_parameter_struct*
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* initialize the struct nor_init_struct */

exmc_norsram_parameter_struct nor_init_struct;

exmc_norsram_struct_para_init (&nor_init_struct);

exmc_norsram_enable

The description of exmc_norsram_enable is shown as below:

Table 3-167. Function exmc_norsram_enable

Function name	exmc_norsram_enable
Function prototype	void exmc_norsram_enable(void);
Function descriptions	enable EXMC NOR/PSRAM bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* enable bank */

exmc_norsram_enable();

exmc_norsram_disable

The description of exmc_norsram_disable is shown as below:

Table 3-168. Function exmc_norsram_disable

Function name	exmc_norsram_disable
Function prototype	void exmc_norsram_disable(void);
Function descriptions	disable EXMC NOR/PSRAM bank
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable bank */

exmc_norsram_disable();

exmc_norsram_page_size_config

The description of exmc_norsram_page_size_config is shown as below:

Table 3-169. Function exmc_norsram_page_size_config

Function name	exmc_norsram_page_size_config
Function prototype	void exmc_norsram_page_size_config(uint32_t page_size);
Function descriptions	configure CRAM page size
Precondition	-
The called functions	-
Input parameter{in}	
page_size	CRAM page size
EXMC_CRAM_AUTO_SPLIT	the clock is generated only during synchronous access
*EXMC_CRAM_PAGE_*	page size is 128 bytes

SIZE_128_BYTES	
EXMC_CRAM_PAGE_SIZE_256_BYTES	page size is 256 bytes
EXMC_CRAM_PAGE_SIZE_512_BYTES	page size is 512 bytes
EXMC_CRAM_PAGE_SIZE_1024_BYTES	page size is 1024 bytes
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CRAM page size */
exmc_norsram_page_size_config (EXMC_CRAM_PAGE_SIZE_128_BYTES);
```

3.10. EXTI

EXTI is the interrupt / event controller in the MCU. It contains up to 19 independent edge detectors and generates interrupt requests or events to the processer. The EXTI registers are listed in chapter *3.10.1*, the EXTI firmware functions are introduced in chapter *3.10.2*.

3.10.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

Table 3-170. EXTI Registers

Registers	Descriptions
EXTI_INTEN	interrupt enable register
EXTI_EVEN	event enable register
EXTI_RTEN	rising edge trigger enable register
EXTI_FTEN	falling edge trigger enable register
EXTI_SWIEV	software interrupt event register
EXTI_PD	pending register

3.10.2. Descriptions of Peripheral functions

EXTI firmware functions are listed in the table shown as below:

Table 3-171. EXTI firmware function

Function name	Function description
exti_deinit	deinitialize the EXTI
exti_init	initialize the EXTI line x

Function name	Function description
exti_interrupt_enable	enable the interrupts from EXTI line x
exti_interrupt_disable	disable the interrupts from EXTI line x
exti_event_enable	enable the events from EXTI line x
exti_event_disable	disable the events from EXTI line x
exti_software_interrupt_enable	enable the software interrupt event from EXTI line x
exti_software_interrupt_disable	disable the software interrupt event from EXTI line x
exti_flag_get	get EXTI line x interrupt pending flag
exti_flag_clear	clear EXTI line x interrupt pending flag
exti_interrupt_flag_get	get EXTI line x interrupt pending flag
exti_interrupt_flag_clear	clear EXTI line x interrupt pending flag

Enum exti_line_enum

Table 3-172. Enum exti_line_enum

Member name	Function description
EXTI_0	EXTI line 0
EXTI_1	EXTI line 1
EXTI_2	EXTI line 2
EXTI_3	EXTI line 3
EXTI_4	EXTI line 4
EXTI_5	EXTI line 5
EXTI_6	EXTI line 6
EXTI_7	EXTI line 7
EXTI_8	EXTI line 8
EXTI_9	EXTI line 9
EXTI_10	EXTI line 10
EXTI_11	EXTI line 11
EXTI_12	EXTI line 12
EXTI_13	EXTI line 13
EXTI_14	EXTI line 14
EXTI_15	EXTI line 15
EXTI_16	EXTI line 16
EXTI_17	EXTI line 17
EXTI_18	EXTI line 18

Enum exti_mode_enum

Table 3-173. Enum exti_mode_enum

Member name	Function description
EXTI_INTERRUPT	EXTI interrupt mode
EXTI_EVENT	EXTI event mode

Enum exti_trig_type_enum

Table 3-174. Enum exti_trig_type_enum

Member name	Function description
EXTI_TRIG_RISING	EXTI rising edge trigger
EXTI_TRIG_FALLING	EXTI falling edge trigger
EXTI_TRIG_BOTH	EXTI rising and falling edge trigger
EXTI_TRIG_NONE	EXTI without rising or falling edge trigger

exti_deinit

The description of exti_deinit is shown as below:

Table 3-175. Function exti_deinit

Function name	exti_deinit
Function prototype	void exti_deinit(void);
Function descriptions	deinitialize the EXTI
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the EXTI */

exti_deinit();
```

exti_init

The description of exti_init is shown as below:

Table 3-176. Function exti_init

Function name	exti_init
Function prototype	void exti_init(exti_line_enum linex, exti_mode_enum mode, exti_trig_type_enum trig_type);
Function descriptions	initialize the EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Input parameter{in}	

mode	EXTI mode, refer to *Table 3-173. Enum exti_mode_enum*
Input parameter{in}	
trig_type	trigger type, refer to *Table 3-174. Enum exti_trig_type_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure EXTI_0 */
exti_init(EXTI_0, EXTI_INTERRUPT, EXTI_TRIG_BOTH);
```

exti_interrupt_enable

The description of exti_interrupt_enable is shown as below:

Table 3-177. Function exti_interrupt_enable

Function name	exti_interrupt_enable
Function prototype	void exti_interrupt_enable(exti_line_enum linex);
Function descriptions	enable the interrupts from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the interrupts from EXTI line 0 */
exti_interrupt_enable(EXTI_0);
```

exti_interrupt_disable

The description of exti_interrupt_disable is shown as below:

Table 3-178. Function exti_interrupt_disable

Function name	exti_interrupt_disable
Function prototype	void exti_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the interrupts from EXTI line x
Precondition	-
The called functions	-

Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the interrupts from EXTI line 0 */
exti_interrupt_disable(EXTI_0);
```

exti_event_enable

The description of exti_event_enable is shown as below:

Table 3-179. Function exti_event_enable

Function name	exti_event_enable
Function prototype	void exti_event_enable(exti_line_enum linex);
Function descriptions	enable the events from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the events from EXTI line 0 */
exti_event_enable(EXTI_0);
```

exti_event_disable

The description of exti_event_disable is shown as below:

Table 3-180. Function exti_event_disable

Function name	exti_event_disable
Function prototype	void exti_event_disable(exti_line_enum linex);
Function descriptions	disable the events from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	

linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the events from EXTI line 0 */
exti_event_disable(EXTI_0);
```

exti_software_interrupt_enable

The description of exti_software_interrupt_enable is shown as below:

Table 3-181. Function exti_software_interrupt_enable

Function name	exti_software_interrupt_enable
Function prototype	void exti_software_interrupt_enable(exti_line_enum linex);
Function descriptions	enable the software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable EXTI line 0 software interrupt */
exti_software_interrupt_enable(EXTI_0);
```

exti_software_interrupt_disable

The description of exti_software_interrupt_disable is shown as below:

Table 3-182. Function exti_software_interrupt_disable

Function name	exti_software_interrupt_disable
Function prototype	void exti_software_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable EXTI line 0 software interrupt */
exti_software_interrupt_disable(EXTI_0);
```

exti_flag_get

The description of exti_flag_get is shown as below:

Table 3-183. Function exti_flag_get

Function name	exti_flag_get
Function prototype	FlagStatus exti_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get EXTI line 0 flag status */
FlagStatus state = exti_flag_get(EXTI_0);
```

exti_flag_clear

The description of exti_flag_clear is shown as below:

Table 3-184. Function exti_flag_clear

Function name	exti_flag_clear
Function prototype	void exti_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* clear EXTI line 0 flag status */

exti_flag_clear(EXTI_0);

exti_interrupt_flag_get

The description of exti_interrupt_flag_get is shown as below:

Table 3-185. Function exti_interrupt_flag_get

Function name	exti_interrupt_flag_get
Function prototype	FlagStatus exti_interrupt_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get EXTI line 0 interrupt flag status */

FlagStatus state = exti_interrupt_flag_get(EXTI_0);

exti_interrupt_flag_clear

The description of exti_interrupt_flag_clear is shown as below:

Table 3-186. Function exti_interrupt_flag_clear

Function name	exti_interrupt_flag_clear
Function prototype	void exti_interrupt_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to *Table 3-172. Enum exti_line_enum*
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* clear EXTI line 0 interrupt flag status */
exti_interrupt_flag_clear(EXTI_0);
```

3.11. FMC

There is flash controller and option byte for GD32E11x series. The FMC registers are listed in chapter *3.11.1* the FMC firmware functions are introduced in chapter *3.11.2*.

3.11.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

Table 3-187. FMC Registers

Registers	Descriptions
FMC_WS	Wait state register
FMC_KEY	Unlock key register
FMC_OBKEY	Option byte unlock key register
FMC_STAT	Status register
FMC_CTL	Control register
FMC_ADDR	Address register
FMC_OBSTAT	Option byte status register
FMC_WP	Erase/Program Protection register
FMC_PID	Product ID register

3.11.2. Descriptions of Peripheral functions

FMC firmware functions are listed in the table shown as below:

Table 3-188. FMC firmware function

Function name	Function description
fmc_wscnt_set	set the FMC wait state counter
fmc_prefetch_enable	enable pre-fetch
fmc_prefetch_disable	disable pre-fetch
fmc_ibus_enable	enable IBUS cache
fmc_ibus_disable	disable IBUS cache
fmc_dbus_enable	enable DBUS cache
fmc_dbus_disable	disable DBUS cache
fmc_ibus_reset	reset IBUS cache

Function name	Function description
fmc_dbus_reset	reset DBUS cache
fmc_program_width_set	set program width to flash memory
fmc_unlock	unlock the main FMC operation
fmc_lock	lock the main FMC operation
fmc_page_erase	FMC erase page
fmc_mass_erase	FMC erase whole chip
fmc_doubleword_program	FMC program a double word at the corresponding address
fmc_word_program	FMC program a word at the corresponding address
ob_unlock	unlock the option byte operation
ob_lock	lock the option byte operation
ob_erase	erase the option byte
ob_write_protection_enable	enable write protection
ob_security_protection_config	configure the option byte security protection
ob_user_write	write the FMC user option byte
ob_data_program	program option bytes data
ob_user_get	get the FMC user option byte
ob_data_get	get OB_DATA in register FMC_OBSTAT
ob_write_protection_get	get the FMC option byte write protection
ob_security_protection_flag_get	get option byte security protection state
fmc_interrupt_enable	enable FMC interrupt
fmc_interrupt_disable	disable FMC interrupt
fmc_flag_get	check flag is set or not
fmc_flag_clear	clear the FMC flag
fmc_interrupt_flag_get	get FMC interrupt flag state
fmc_interrupt_flag_clear	clear FMC interrupt flag state
fmc_state_get	get FMC state
fmc_ready_wait	check FMC ready or not

Enum fmc_state_enum

Table 3-189. Enum fmc_state_enum

enum name	enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_PGERR	program error
FMC_PGAERR	program alignment error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error

fmc_wscnt_set

The description of fmc_wscnt_set is shown as below:

Table 3-190. Function fmc_wscnt_set

Function name	fmc_wscnt_set
Function prototype	void fmc_wscnt_set(uint32_t wscnt);
Function descriptions	set the wait state counter value
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
wscnt	wait state counter value
FMC_WAIT_STATE_0	FMC 0 wait
FMC_WAIT_STATE_1	FMC 1 wait
FMC_WAIT_STATE_2	FMC 2 wait
FMC_WAIT_STATE_3	FMC 3 wait
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the wait state counter value */
fmc_wscnt_set (FMC_WAIT_STATE_1);
```

fmc_prefetch_enable

The description of fmc_prefetch_enable is shown as below:

Table 3-191. Function fmc_prefetch_enable

Function name	fmc_prefetch_enable
Function prototype	void fmc_prefetch_enable(void);
Function descriptions	enable pre-fetch
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable pre-fetch */
fmc_prefetch_enable( );
```

fmc_prefetch_disable

The description of fmc_prefetch_disable is shown as below:

Table 3-192. Function fmc_prefetch_disable

Function name	fmc_prefetch_disable
Function prototype	void fmc_prefetch_disable (void);
Function descriptions	disable pre-fetch
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable pre-fetch */

fmc_prefetch_disable( );
```

fmc_ibus_enable

The description of fmc_ibus_enable is shown as below:

Table 3-193. Function fmc_ibus_enable

Function name	fmc_ibus_enable
Function prototype	void fmc_ibus_enable(void);
Function descriptions	enable IBUS cache
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable IBUS cache */

fmc_ibus_enable( );
```

fmc_ibus_disable

The description of fmc_ibus_disable is shown as below:

Table 3-194. Function fmc_ibus_disable

Function name	fmc_ibus_disable
Function prototype	void fmc_ibus_disable(void);
Function descriptions	disable IBUS cache
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable IBUS cache */

fmc_ibus_disable( );
```

fmc_dbus_enable

The description of fmc_dbus_enable is shown as below:

Table 3-195. Function fmc_dbus_enable

Function name	fmc_dbus_enable
Function prototype	void fmc_dbus_enable(void);
Function descriptions	enable DBUS cache
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DBUS cache */

fmc_dbus_enable( );
```

fmc_dbus_disable

The description of fmc_dbus_disable is shown as below:

Table 3-196. Function fmc_dbus_disable

Function name	fmc_dbus_disable
Function prototype	void fmc_dbus_disable(void);
Function descriptions	disable DBUS cache
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DBUS cache */

fmc_dbus_disable( );
```

fmc_ibus_reset

The description of fmc_ibus_reset is shown as below:

Table 3-197. Function fmc_ibus_reset

Function name	fmc_ibus_reset
Function prototype	void fmc_ibus_reset (void);
Function descriptions	reset IBUS cache
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset IBUS cache */

fmc_ibus_reset( );
```

fmc_dbus_reset

The description of fmc_dbus_reset is shown as below:

Table 3-198. Function fmc_dbus_reset

Function name	fmc_dbus_reset
Function prototype	void fmc_dbus_reset(void);
Function descriptions	reset DBUS cache
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset DBUS cache */

fmc_dbus_reset( );
```

fmc_program_width_set

The description of fmc_program_width_set is shown as below:

Table 3-199. Function fmc_program_width_set

Function name	fmc_program_width_set
Function prototype	void fmc_program_width_set(uint32_t pgw);
Function descriptions	set program width to flash memory
Precondition	-
The called functions	-
Input parameter{in}	
pgw	program width
FMC_PROG_W_32B	32-bit program width to flash memory
FMC_PROG_W_64B	64-bit program width to flash memory
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set program width to flash memory */

fmc_program_width_set(FMC_PROG_W_32B);
```

fmc_unlock

The description of fmc_unlock is shown as below:

Table 3-200. Function fmc_unlock

Function name	fmc_unlock
Function prototype	void fmc_unlock (void);
Function descriptions	unlock the main FMC operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the main FMC operation */
fmc_unlock ( );
```

fmc_lock

The description of fmc_lock is shown as below:

Table 3-201. Function fmc_lock

Function name	fmc_lock
Function prototype	void fmc_lock(void);
Function descriptions	lock the main FMC operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the main FMC operation */
fmc_lock( );
```

fmc_page_erase

The description of fmc_page_erase is shown as below:

Table 3-202. Function fmc_page_erase

Function name	fmc_page_erase
Function prototype	fmc_state_enum fmc_page_erase(uint32_t page_address);
Function descriptions	erase page
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
page_address	the page address to be erased
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

```
/* erase page */

fmc_page_erase ( 0x08004000);
```

fmc_mass_erase

The description of fmc_mass_erase is shown as below:

Table 3-203. Function fmc_mass_erase

Function name	fmc_mass_erase
Function prototype	fmc_state_enum fmc_mass_erase(void);
Function descriptions	erase whole chip
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

```
/* erase whole chip */

fmc_mass_erase ( );
```

fmc_doubleword_program

The description of fmc_doubleword_program is shown as below:

Table 3-204. Function fmc_doubleword_program

Function name	fmc_doubleword_program
Function prototype	fmc_state_enum fmc_doubleword_program(uint32_t address, uint64_t data);
Function descriptions	program a double word at the corresponding address
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
address	the address to program
data	the data to program
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

```
/* program a double word at the corresponding address */
fmc_doubleword_program( 0x08004000,0xaabbccddeeffgghh);
```

fmc_word_program

The description of fmc_word_program is shown as below:

Table 3-205. Function fmc_word_program

Function name	fmc_word_program
Function prototype	fmc_state_enum fmc_word_program(uint32_t address, uint32_t data);
Function descriptions	program a word at the corresponding address
Precondition	fmc_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
address	the address to program
data	the data to program
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

```
/* program a word at the corresponding address */
```

fmc_word_program (0x08004000,0xaabbccdd);

ob_unlock

The description of ob_unlock is shown as below:

Table 3-206. Function ob_unlock

Function name	ob_unlock
Function prototype	void ob_unlock(void);
Function descriptions	unlock the option byte operation
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the option byte operation */

ob_unlock( );
```

ob_lock

The description of ob_lock is shown as below:

Table 3-207. Function ob_lock

Function name	ob_lock
Function prototype	void ob_lock(void);
Function descriptions	lock the option byte operation
Precondition	fmc_lock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the option byte operation */

ob_lock( );
```

ob_erase

The description of ob_erase is shown as below:

Table 3-208. Function ob_erase

Function name	ob_erase
Function prototype	void ob_erase(void);
Function descriptions	erase the FMC option byte
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* erase the FMC option byte */
ob_ erase ( );
```

ob_write_protection_enable

The description of ob_write_protection_enable is shown as below:

Table 3-209. Function ob_write_protection_enable

Function name	ob_write_protection_enable
Function prototype	fmc_state_enum ob_write_protection_enable(uint32_t ob_wp);
Function descriptions	enable write protection
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
ob_wp	enable write protection
OB_WP_x	write protect specify sector x
OB_WP_ALL	write protect all sector
Output parameter{out}	
-	-
Return value	
fmc_state_enum	*state of FMC*

Example:

```
/* enable write protection */
ob_write_protection_enable (OB_WP_7);
```

ob_security_protection_config

The description of ob_security_protection_config is shown as below:

Table 3-210. Function ob_security_protection_config

Function name	ob_security_protection_config
Function prototype	fmc_state_enum ob_security_protection_config (uint8_t ob_spc);
Function descriptions	configure security protection
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
ob_spc	specify security protection
FMC_NSPC	no security protection
FMC_USPC	under security protection
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

```
/* enable security protection */
ob_security_protection_config (FMC_USPC);
```

ob_user_write

The description of ob_user_write is shown as below:

Table 3-211. Function ob_user_write

Function name	ob_user_write
Function prototype	fmc_state_enum ob_user_write(uint8_t ob_fwdgt, uint8_t ob_deepsleep, uint8_t ob_stdby);
Function descriptions	program the FMC user option byte
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
ob_fwdgt	option byte watchdog value
OB_FWDGT_SW	software free watchdog
OB_FWDGT_HW	hardware free watchdog
Input parameter{in}	
ob_deepsleep	option byte deepsleep reset value
OB_DEEPSLEEP_NRST	no reset when entering deepsleep mode
OB_DEEPSLEEP_RST	generate a reset instead of entering deepsleep mode
Input parameter{in}	

ob_stdby	option byte standby reset value
OB_STDBY_NRST	no reset when entering standby mode
OB_STDBY_RST	generate a reset instead of entering standby mode
Output parameter{out}	
-	-
Return value	
fmc_state_enum	*state of FMC*

Example:

```
/* configure user option byte */
ob_user_write(OB_FWDGT_HW, OB_DEEPSLEEP_RST, OB_STDBY_RST);
```

ob_data_program

The description of ob_data_program is shown as below:

Table 3-212. Function ob_data_program

Function name	ob_data_program
Function prototype	fmc_state_enum ob_data_program(uint32_t address, uint8_t data);
Function descriptions	program option bytes data
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
address	the option bytes address to be programmed
data	the byte to be programmed
Output parameter{out}	
-	-
Return value	
fmc_state_enum	*state of FMC*

Example:

```
/* program option bytes data */
ob_data_program (0x1ffff804, 0x56);
```

ob_user_get

The description of ob_user_get is shown as below:

Table 3-213. Function ob_user_get

Function name	ob_user_get
Function prototype	uint8_t ob_user_get(void);
Function descriptions	get the FMC user option byte
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the FMC user option byte values(0xF0 - 0xFF)

Example:

```
/* get the FMC user option byte */
uint8_t user = ob_user_get ( );
```

ob_data_get

The description of ob_data_get is shown as below:

Table 3-214. Function ob_data_program

Function name	ob_data_get
Function prototype	Uint16_t ob_data_get(void);
Function descriptions	get the FMC data option byte
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
Uint16_t	the FMC data option byte values(0x0 - 0xFFFF)

Example:

```
/* get the FMC data option byte */
Uint16_t data = ob_data_get ( );
```

ob_write_protection_get

The description of ob_ write_protection_get is shown as below:

Table 3-215. Function ob_write_protection_get

Function name	ob_ write_protection_get
Function prototype	uint32_t ob_write_protection_get(void);
Function descriptions	get the FMC option byte write protection
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the FMC write protection option byte value(0x0 - 0XFFFFFFFF)

Example:

```
/* get the FMC option byte write protection */
uint32_t wp = ob_write_protection_get ( );
```

ob_security_protection_flag_get

The description of ob_security_protection_flag_get is shown as below:

Table 3-216. Function ob_security_protection_flag_get

Function name	ob_security_protection_flag_get
Function prototype	FlagStatus ob_security_protection_flag_get(void);
Function descriptions	get the FMC option byte security protection
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the FMC option byte security protection */
FlagStatus spc = ob_security_protection_flag_get( );
```

fmc_interrupt_enable

The description of fmc_interrupt_enable is shown as below:

Table 3-217. Function fmc_interrupt_enable

Function name	fmc_interrupt_enable
Function prototype	void fmc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable FMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	

interrupt	the FMC interrupt source
FMC_INT_END	FMC end of program interrupt
FMC_INT_ERR	FMC error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable FMC interrupt */

fmc_interrupt_enable(FMC_INT_END);
```

fmc_interrupt_disable

The description of fmc_interrupt_disable is shown as below:

Table 3-218. Function fmc_interrupt_disable

Function name	fmc_interrupt_disable
Function prototype	void fmc_interrupt_disable(uint32_t interrupt);
Function descriptions	disable FMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the FMC interrupt source
FMC_INT_END	FMC end of program interrupt
FMC_INT_ERR	FMC error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable FMC interrupt */

fmc_interrupt_disable(FMC_INT_END);
```

fmc_flag_get

The description of fmc_flag_get is shown as below:

Table 3-219. Function fmc_flag_get

Function name	fmc_flag_get
Function prototype	FlagStatus fmc_flag_get(uint32_t flag);
Function descriptions	check flag is set or not

Precondition	-
The called functions	-
Input parameter{in}	
flag	check FMC flag
FMC_FLAG_BUSY	FMC busy flag bit
FMC_FLAG_PGERR	FMC operation error flag bit
FMC_FLAG_PGAERR	FMC program alignment error flag bit
FMC_FLAG_WPERR	FMC erase/program protection error flag bit
FMC_FLAG_END	FMC end of operation flag bit
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get FMC flag */
FlagStatus flag = fmc_flag_get(FMC_FLAG_END);
```

fmc_flag_clear

The description of fmc_flag_clear is shown as below:

Table 3-220. Function fmc_flag_clear

Function name	fmc_flag_clear
Function prototype	void fmc_flag_clear(uint32_t flag);
Function descriptions	clear the FMC flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	clear FMC flag
FMC_FLAG_PGERR	FMC operation error flag bit
FMC_FLAG_PGAERR	FMC program alignment error flag bit
FMC_FLAG_WPERR	FMC erase/program protection error flag bit
FMC_FLAG_END	FMC end of operation flag bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* get FMC flag */
fmc_flag_clear(FMC_FLAG_END);
```

fmc_interrupt_flag_get

The description of fmc_interrupt_flag_get is shown as below:

Table 3-221. Function fmc_interrupt_flag_get

Function name	fmc_interrupt_flag_get
Function prototype	FlagStatus fmc_interrupt_flag_get(fmc_interrupt_flag_enum flag);
Function descriptions	get FMC interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	FMC flag
FMC_INT_FLAG_PGERR	FMC operation error flag bit
FMC_INT_FLAG_PGAERR	FMC program alignment error flag bit
FMC_INT_FLAG_WPERR	FMC erase/program protection error flag bit
FMC_INT_FLAG_END	FMC end of operation flag bit
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get FMC flag */
FlagStatus flag = fmc_interrupt_flag_get (FMC_INT_FLAG_PGERR);
```

fmc_interrupt_flag_clear

The description of fmc_interrupt_flag_get is shown as below:

Table 3-222. Function fmc_interrupt_flag_clear

Function name	fmc_interrupt_flag_ clear
Function prototype	FlagStatus fmc_interrupt_flag_ clear (fmc_interrupt_flag_enum flag);
Function descriptions	clear FMC interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	clear FMC flag
FMC_INT_FLAG_PGERR	FMC operation error flag bit
FMC_INT_FLAG_PGAERR	FMC program alignment error flag bit

FMC_INT_FLAG_WPERR	FMC erase/program protection error flag bit
FMC_INT_FLAG_END	FMC end of operation flag bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear FMC flag */
fmc_interrupt_flag_get (FMC_INT_FLAG_PGERR);
```

fmc_state_get

The description of fmc_state_get is shown as below:

Table 3-223. Function fmc_state_get

Function name	fmc_state_get
Function prototype	fmc_state_enum fmc_state_get(void);
Function descriptions	get the FMC state
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	***state of FMC***

Example:

```
/* get the FMC state */
fmc_state_enum state = fmc_state_get( );
```

fmc_ready_wait

The description of fmc_ready_waitis shown as below:

Table 3-224. Function fmc_ready_wait

Function name	fmc_ready_wait
Function prototype	fmc_state_enum fmc_ready_wait(uint32_t timeout);
Function descriptions	check whether FMC is ready or not
Precondition	-
The called functions	fmc_state_get()

Input parameter{in}	
timeout	count of loop
Output parameter{out}	
-	-
Return value	
fmc_state_enum	*state of FMC*

Example:

```
/* check whether FMC is ready or not */
fmc_ready_wait (0x00001000 );
```

3.12. FWDGT

The free watchdog timer (FWDGT) is a hardware timing circuitry that can be used to detect system failures due to software malfunctions. It's suitable for the situation that requires an independent environment and lower timing accuracy.The FWDGT registers are listed in chapter *3.12.1* the FWDGT firmware functions are introduced in chapter *3.12.2*.

3.12.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

Table 3-225. FWDGT Registers

Registers	Descriptions
FWDGT_CTL	Control register
FWDGT_PSC	Prescaler register
FWDGT_RLD	Reload register
FWDGT_STAT	Status register

3.12.2. Descriptions of Peripheral functions

FWDGT firmware functions are listed in the table shown as below:

Table 3-226. FWDGT firmware function

Function name	Function description
fwdgt_write_enable	enable write access to FWDGT_PSC and FWDGT_RLD
fwdgt_write_disable	disable write access to FWDGT_PSC and FWDGT_RLD
fwdgt_enable	start the free watchdog timer counter
fwdgt_prescaler_value_config	configure the free watchdog timer counter prescaler value
fwdgt_reload_value_config	configure the free watchdog timer counter reload value
fwdgt_counter_reload	reload the counter of FWDGT
fwdgt_config	configure counter reload value, and prescaler divider value
fwdgt_flag_get	get flag state of FWDGT

fwdgt_write_enable

The description of fwdgt_write_enable is shown as below:

Table 3-227. Function fwdgt_write_ensable

Function name	fwdgt_write_enable
Function prototype	void fwdgt_write_enable(void);
Function descriptions	enable write access to FWDGT_PSC and FWDGT_RLD
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable write access to FWDGT_PSC and FWDGT_RLD */
fwdgt_write_enable ( );
```

fwdgt_write_disable

The description of fwdgt_write_disable is shown as below:

Table 3-228. Function fwdgt_write_disable

Function name	fwdgt_write_disable
Function prototype	void fwdgt_write_disable(void);
Function descriptions	disable write access to FWDGT_PSC and FWDGT_RLD
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable write access to FWDGT_PSC and FWDGT_RLD */
fwdgt_write_disable ( );
```

fwdgt_enable

The description of fwdgt_enable is shown as below:

Table 3-229. Function fwdgt_enable

Function name	fwdgt_enable
Function prototype	void fwdgt_enable(void);
Function descriptions	start the free watchdog timer counter
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start the free watchdog timer counter */

fwdgt_enable ( );
```

fwdgt_prescaler_value_config

The description of fwdgt_prescaler_value_config is shown as below:

Table 3-230. Function fwdgt_prescaler_value_config

Function name	fwdgt_prescaler_value_config
Function prototype	ErrStatus fwdgt_prescaler_value_config(uint16_t prescaler_value);
Function descriptions	configure the free watchdog timer counter prescaler value
Precondition	-
Input parameter{in}	
prescaler_value	specify prescaler value
FWDGT_PSC_DIV4	FWDGT prescaler set to 4
FWDGT_PSC_DIV8	FWDGT prescaler set to 8
FWDGT_PSC_DIV16	FWDGT prescaler set to 16
FWDGT_PSC_DIV32	FWDGT prescaler set to 32
FWDGT_PSC_DIV64	FWDGT prescaler set to 64
FWDGT_PSC_DIV128	FWDGT prescaler set to 128
FWDGT_PSC_DIV256	FWDGT prescaler set to 256
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```
/* set FWDGT prescaler to 256 */

ErrStatus flag;

flag = fwdgt_prescaler_value_config (FWDGT_PSC_DIV256);
```

fwdgt_reload_value_config

The description of fwdgt_reload_value_config is shown as below:

Table 3-231. Function fwdgt_reload_value_config

Function name	fwdgt_reload_value_config
Function prototype	ErrStatus fwdgt_reload_value_config(uint16_t reload_value);
Function descriptions	configure the free watchdog timer counter reload value
Precondition	-
Input parameter{in}	
reload_value	reload_value: specify window value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```
/* set FWDGT reload value to 0x0FFF */

ErrStatus flag;

flag = fwdgt_reload_value_config (0x0FFF);
```

fwdgt_counter_reload

The description of fwdgt_counter_reload is shown as below:

Table 3-232. Function fwdgt_counter_reload

Function name	fwdgt_counter_reload
Function prototype	void fwdgt_counter_reload(void);
Function descriptions	reload the counter of FWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reload FWDGT counter */

fwdgt_counter_reload ( );
```

fwdgt_config

The description of fwdgt_config is shown as below:

Table 3-233. Function fwdgt_config

Function name	fwdgt_config
Function prototype	ErrStatus fwdgt_config(uint16_t reload_value, uint8_t prescaler_div);
Function descriptions	configure counter reload value, and prescaler divider value
Precondition	-
The called functions	-
Input parameter{in}	
reload_value	specify reload value(0x0000 - 0x0FFF)-
Input parameter{in}	
prescaler_div	FWDGT prescaler value-
FWDGT_PSC_DIV4	FWDGT prescaler set to 4
FWDGT_PSC_DIV8	FWDGT prescaler set to 8
FWDGT_PSC_DIV16	FWDGT prescaler set to 16
FWDGT_PSC_DIV32	FWDGT prescaler set to 32
FWDGT_PSC_DIV64	FWDGT prescaler set to 64
FWDGT_PSC_DIV128	FWDGT prescaler set to 128
FWDGT_PSC_DIV256	FWDGT prescaler set to 256
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* confiure FWDGT counter clock: 40KHz(IRC40K) / 64 = 0.625 KHz */

fwdgt_config(2*500, FWDGT_PSC_DIV64);
```

fwdgt_flag_get

The description of fwdgt_flag_get is shown as below:

Table 3-234. Function fwdgt_flag_get fwdgt_write_disable

Function name	fwdgt_flag_get
Function prototype	FlagStatus fwdgt_flag_get(uint16_t flag);
Function descriptions	get flag state of FWDGT
Precondition	-

The called functions	-
Input parameter{in}	
flag	flag to get
FWDGT_FLAG_PUD	a write operation to FWDGT_PSC register is on going
FWDGT_FLAG_RUD	a write operation to FWDGT_RLD register is on going
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* test if a prescaler value update is on going */
FlagStatus status;
status = fwdgt_flag_get (FWDGT_FLAG_PUD);
if(status == RESET)
{
...
}else
{
...
}
```

3.13. GPIO

GPIO is used to implement logic input/output functions for the devices. The GPIO registers are listed in chapter *3.13.1*, the GPIO firmware functions are introduced in chapter *3.13.2*.

3.13.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

Table 3-235. GPIO Registers

Registers	Descriptions
GPIOx_CTL0	Port control register 0
GPIOx_CTL1	Port control register 1
GPIOx_ISTAT	Port input status register
GPIOx_OCTL	Port output control register
GPIOx_BOP	Port bit operate register
GPIOx_BC	Port bit clear register

Registers	Descriptions
GPIOx_LOCK	Port configuration lock register
GPIOx_SPD	Port bit speed register
AFIO_EC	Event control register
AFIO_PCF0	AFIO port configuration register 0
AFIO_EXTISS0	EXTI sources selection register 0
AFIO_EXTISS1	EXTI sources selection register 1
AFIO_EXTISS2	EXTI sources selection register 2
AFIO_EXTISS3	EXTI sources selection register 3
AFIO_PCF1	AFIO port configuration register 1
AFIO_CPSCTL	IO compensation control register

3.13.2. Descriptions of Peripheral functions

GPIO firmware functions are listed in the table shown as below:

Table 3-236. GPIO firmware function

Function name	Function description
gpio_deinit	reset GPIO port
gpio_afio_deinit	reset alternate function I/O(AFIO)
gpio_init	GPIO parameter initialization
gpio_bit_set	set GPIO pin
gpio_bit_reset	reset GPIO pin
gpio_bit_write	write data to the specified GPIO pin
gpio_port_write	write data to the specified GPIO port
gpio_input_bit_get	get GPIO pin input status
gpio_input_port_get	get GPIO port input status
gpio_output_bit_get	get GPIO pin output status
gpio_output_port_get	get GPIO port output status
gpio_pin_remap_config	configure GPIO pin remap
gpio_exti_source_select	select GPIO pin exti sources
gpio_event_output_config	configure GPIO pin event output
gpio_event_output_enable	enable GPIO pin event output
gpio_event_output_disable	disable GPIO pin event output
gpio_pin_lock	lock GPIO pin
gpio_compensation_config	configure the I/O compensation cell
gpio_compensation_flag_get	check the I/O compensation cell is ready or not

gpio_deinit

The description of gpio_deinit is shown as below:

Table 3-237. Function gpio_deinit

Function name	gpio_deinit

Function prototype	void gpio_deinit(uint32_t gpio_periph);
Function descriptions	reset GPIO port
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset GPIOA */
gpio_deinit (GPIOA);
```

gpio_afio_deinit

The description of gpio_afio_deinit is shown as below:

Table 3-238. Function gpio_afio_deinit

Function name	gpio_afio_deinit
Function prototype	void gpio_afio_deinit(void);
Function descriptions	reset alternate function I/O(AFIO)
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset alternate function */
gpio_afio_deinit();
```

gpio_init

The description of gpio_init is shown as below:

Table 3-239. Function gpio_init

Function name	gpio_init

Function prototype	void gpio_init(uint32_t gpio_periph, uint32_t mode, uint32_t speed, uint32_t pin);
Function descriptions	GPIO parameter initialization
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Input parameter{in}	
gpio_mode	gpio pin mode
GPIO_MODE_AIN	analog input mode
GPIO_MODE_IN_FLOATING	floating input mode
GPIO_MODE_IPD	pull-down input mode
GPIO_MODE_IPU	pull-up input mode
GPIO_MODE_OUT_OD	GPIO output with open-drain
GPIO_MODE_OUT_PP	GPIO output with push-pull
GPIO_MODE_AF_OD	AFIO output with open-drain
GPIO_MODE_AF_PP	AFIO output with push-pull
Input parameter{in}	
speed	gpio output max speed value
GPIO_OSPEED_10MHZ	output max speed 10MHz
GPIO_OSPEED_2MHZ	output max speed 2MHz
GPIO_OSPEED_50MHZ	output max speed 50MHz
GPIO_OSPEED_MAX	output max speed more than 50MHz
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x = 0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* config PA0 as analog input mode */

gpio_init (GPIOA, GPIO_MODE_AIN, GPIO_OSPEED_50MHZ, GPIO_PIN_0);

gpio_bit_set

The description of gpio_bit_set is shown as below:

Table 3-240. Function gpio_bit_set

Function name	gpio_bit_set
Function prototype	void gpio_bit_set(uint32_t gpio_periph,uint32_t pin);
Function descriptions	set GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x = 0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set PA0 */

gpio_bit_set (GPIOA, GPIO_PIN_0);
```

gpio_bit_reset

The description of gpio_bit_reset is shown as below:

Table 3-241. Function gpio_bit_reset

Function name	gpio_bit_reset
Function prototype	void gpio_bit_reset(uint32_t gpio_periph,uint32_t pin);
Function descriptions	reset GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x = 0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* reset PA0 */
gpio_bit_set (GPIOA, GPIO_PIN_0);
```

gpio_bit_write

The description of gpio_bit_write is shown as below:

Table 3-242. Function gpio_bit_write

Function name	gpio_bit_write
Function prototype	void gpio_bit_write(uint32_t gpio_periph,uint32_t pin,bit_status bit_value);
Function descriptions	write data to the specified GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x = 0..15)
GPIO_PIN_ALL	All pins
Input parameter{in}	
bit_value	SET or RESET
RESET	clear the port pin
SET	set the port pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write 1 to PA0 */
gpio_bit_write (GPIOA, GPIO_PIN_0, SET);
```

gpio_port_write

The description of gpio_port_write is shown as below:

Table 3-243. Function gpio_port_write

Function name	gpio_port_write
Function prototype	void gpio_port_write(uint32_t gpio_periph,uint16_t data);
Function descriptions	write data to the specified GPIO port
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Input parameter{in}	
data	specify the value to be written to the port output data register
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*write 1010 0101 to Port A */

gpio_port_write (GPIOA, 0xA5);
```

gpio_input_bit_get

The description of gpio_input_bit_get is shown as below:

Table 3-244. Function gpio_input_bit_get

Function name	gpio_input_bit_get
Function prototype	FlagStatus gpio_input_bit_get(uint32_t gpio_periph,uint32_t pin);
Function descriptions	get GPIO pin input status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x = 0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get status of PA0 */

FlagStatus bit_state;

bit_state = gpio_input_bit_get (GPIOA, GPIO_PIN_0);
```

gpio_input_port_get

The description of gpio_input_port_get is shown as below:

Table 3-245. Function gpio_input_port_get

Function name	gpio_input_port_get
Function prototype	uint16_t gpio_input_port_get(uint32_t gpio_periph);
Function descriptions	get GPIO port input status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Output parameter{out}	
-	-
Return value	
uint16_t	0x00 - 0xFF

Example:

```
/* get input value of Port A */

uint16_t port_state;

port_state = gpio_input_bit_get (GPIOA);
```

gpio_output_bit_get

The description of gpio_output_bit_get is shown as below:

Table 3-246. Function gpio_output_bit_get

Function name	gpio_output_bit_get
Function prototype	FlagStatus gpio_output_bit_get(uint32_t gpio_periph,uint32_t pin);
Function descriptions	get GPIO pin output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Input parameter{in}	
pin	GPIO pin

GPIO_PIN_x	GPIO_PIN_x(x = 0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get output status of PA0 */
FlagStatus bit_state;
bit_state = gpio_output_bit_get (GPIOA, GPIO_PIN_0);
```

gpio_output_port_get

The description of gpio_output_port_get is shown as below:

Table 3-247. Function gpio_output_port_get

Function name	gpio_output_port_get
Function prototype	uint16_t gpio_output_port_get(uint32_t gpio_periph);
Function descriptions	get GPIO port output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Output parameter{out}	
-	-
Return value	
Uint16_t	0x00 - 0xFF

Example:

```
/* get output value of Port A */
uint16_t port_state;
port_state = gpio_output_port_get (GPIOA);
```

gpio_pin_remap_config

The description of gpio_pin_remap_config is shown as below:

Table 3-248. Function gpio_pin_remap_config

Function name	gpio_pin_remap_config
Function prototype	void gpio_pin_remap_config(uint32_t remap, ControlStatus newvalue);

Function descriptions	configure GPIO pin remap
Precondition	-
The called functions	-
Input parameter{in}	
gpio_remap	select the pin to remap
GPIO_SPI0_REMAP	SPI0 remapping
GPIO_I2C0_REMAP	I2C0 remapping
GPIO_USART0_REMAP	USART0 remapping
GPIO_USART1_REMAP	USART1 remapping
GPIO_USART2_PARTIAL_REMAP	USART2 partial remapping
GPIO_USART2_FULL_REMAP	USART2 full remapping
GPIO_TIMER0_PARTIAL_REMAP	TIMER0 partial remapping
GPIO_TIMER0_FULL_REMAP	TIMER0 full remapping
GPIO_TIMER1_PARTIAL_REMAP1	TIMER1 partial remapping
GPIO_TIMER1_PARTIAL_REMAP2	TIMER1 partial remapping
GPIO_TIMER1_FULL_REMAP	TIMER1 full remapping
GPIO_TIMER2_PARTIAL_REMAP	TIMER2 partial remapping
GPIO_TIMER2_FULL_REMAP	TIMER2 full remapping
GPIO_TIMER3_REMAP	TIMER3 remapping
GPIO_PD01_REMAP	PD01 remapping
GPIO_ADC0_ETRGRT_REMAP	ADC0 external trigger routine conversion remapping
GPIO_ADC1_ETRGRT_REMAP	ADC1 external trigger routine conversion remapping
GPIO_TIMER4CH3_IREMAP	TIMER4 channel3 internal remapping
GPIO_SWJ_NONJTRST_REMAP	full SWJ(JTAG-DP + SW-DP),but without NJTRST
GPIO_SWJ_SWDPENABLE_REMAP	JTAG-DP disabled and SW-DP enabled
*GPIO_SWJ_DISABLE_*	JTAG-DP disabled and SW-DP disabled

REMAP	
GPIO_SPI2_REMAP	SPI2 remapping
GPIO_TIMER1ITR0_REMAP	TIMER1 internal trigger 0 remapping
GPIO_TIMER8_REMAP	TIMER8 remapping
GPIO_EXMC_NADV_REMAP	EXMC_NADV connect/disconnect
GPIO_CTC_REMAP0	CTC remapping(PD15)
Input parameter{in}	
newvalue	ENABLE / DISABLE
ENABLE	
DISABLE	
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable SPI0 remapping */

gpio_pin_remap_config (GPIO_SPI0_REMAP, ENABLE);

gpio_exti_source_select

The description of gpio_exti_source_select is shown as below:

Table 3-249. Function gpio_exti_source_select

Function name	gpio_exti_source_select
Function prototype	void gpio_exti_source_select(uint8_t output_port, uint8_t output_pin);
Function descriptions	select GPIO pin exti sources
Precondition	-
The called functions	-
Input parameter{in}	
gpio_outputport	gpio event output port
GPIO_PORT_SOURCE_GPIOx	output port source (x= A,B,C,D,E)
Input parameter{in}	
gpio_outputpin	gpio event output pin
GPIO_PIN_SOURCE_x	Pin number(x = 0..15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* config PA0 as EXTI source */

gpio_exti_source_select (GPIO_PORT_SOURCE_GPIOA, GPIO_PIN_SOURCE_0);

gpio_event_output_config

The description of gpio_event_output_config is shown as below:

Table 3-250. Function gpio_event_output_config

Function name	gpio_event_output_config
Function prototype	void gpio_event_output_config(uint8_t output_port, uint8_t output_pin);
Function descriptions	configure GPIO pin event output
Precondition	-
The called functions	-
Input parameter{in}	
gpio_outputport	gpio event output port
GPIO_EVENT_PORT_GPIOx	event output port x (x= A,B,C,D,E)
Input parameter{in}	
gpio_outputpin	gpio event output pin
GPIO_EVENT_PIN_x	Pin number (x = 0..15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* Config PA0 as the output of event */

gpio_event_output_config (GPIO_EVENT_PORT_GPIOA, GPIO_EVENT_PIN_0);

gpio_event_output_enable

The description of gpio_event_output_enable is shown as below:

Table 3-251. Function gpio_event_output_enable

Function name	gpio_event_output_enable
Function prototype	void gpio_event_output_enable(void);
Function descriptions	enable GPIO pin event output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* enable GPIO pin event output */

gpio_event_output_enable(void);

gpio_event_output_disable

The description of gpio_event_output_disable is shown as below:

Table 3-252. Function gpio_event_output_disable

Function name	gpio_event_output_disable
Function prototype	void gpio_event_output_disable(void);
Function descriptions	disable GPIO pin event output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable GPIO pin event output */

gpio_event_output_disable(void);

gpio_pin_lock

The description of gpio_pin_lock is shown as below:

Table 3-253. Function gpio_pin_lock

Function name	gpio_pin_lock
Function prototype	void gpio_pin_lock(uint32_t gpio_periph,uint32_t pin);
Function descriptions	lock GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E)
Input parameter{in}	

pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x = 0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock PA0 */
gpio_pin_lock (GPIOA, GPIO_PIN_0);
```

gpio_compensation_config

The description of gpio_compensation_config is shown as below:

Table 3-254. Function gpio_compensation_config

Function name	gpio_compensation_config
Function prototype	void gpio_compensation_config(uint32_t compensation);
Function descriptions	configure the I/O compensation cell
Precondition	-
The called functions	-
Input parameter{in}	
compensation	specifies the I/O compensation cell mode
GPIO_COMPENSATION_ENABLE	I/O compensation cell is enabled
GPIO_COMPENSATION_DISABLE	I/O compensation cell is disabled
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enabled I/O compensation cell */
gpio_compensation_config (GPIO_COMPENSATION_ENABLE);
```

gpio_compensation_flag_get

The description of gpio_compensation_flag_get is shown as below:

Table 3-255. Function gpio_compensation_flag_get

Function name	gpio_compensation_flag_get

Function prototype	FlagStatus gpio_compensation_flag_get(void);
Function descriptions	check the I/O compensation cell is ready or not
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check the I/O compensation cell state */
FlagStatus cell_state;
cell_state = gpio_compensation_flag_get (void);
```

3.14. I2C

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. The I2C registers are listed in chapter *3.14.1*, the I2C firmware functions are introduced in chapter *3.14.2*

3.14.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

Table 3-256. I2C Registers

Registers	Descriptions
I2C_CTL0	Control register 0
I2C_CTL1	Control register 1
I2C_SADDR0	Slave address register 0
I2C_SADDR1	Slave address register 1
I2C_DATA	Transfer buffer register
I2C_STAT0	Transfer status register 0
I2C_STAT1	Transfer status register 1
I2C_CKCFG	Clock configure register
I2C_RT	Rise time register
I2C_SAMCS	SAM control and status register
I2C_FMPCFG	Fast mode plus configure register

3.14.2. Descriptions of Peripheral functions

I2C firmware functions are listed in the table shown as below:

Table 3-257. I2C firmware function

Function name	Function description
i2c_deinit	reset I2C
i2c_clock_config	configure I2C clock
i2c_mode_addr_config	configure I2C address
i2c_smbus_type_config	select SMBus type
i2c_ack_config	whether or not to send an ACK
i2c_ackpos_config	configure I2C POAP position
i2c_master_addressing	master sends slave address
i2c_dualaddr_enable	enable dual-address mode
i2c_dualaddr_disable	disable dual-address mode
i2c_enable	enable I2C
i2c_disable	disable I2C
i2c_start_on_bus	generate a START condition on I2C bus
i2c_stop_on_bus	generate a STOP condition on I2C bus
i2c_data_transmit	I2C transmit data function
i2c_data_receive	I2C receive data function
i2c_dma_config	configure I2C DMA mode
i2c_dma_last_transfer_config	configure whether next DMA EOT is DMA last transfer or not
i2c_stretch_scl_low_config	whether to stretch SCL low when data is not ready in slave mode
i2c_slave_response_to_gcall_config	whether or not to response to a general call
i2c_software_reset_config	configure software reset of I2C
i2c_pec_config	configure I2C PEC calculation
i2c_pec_transfer_config	configure whether to transfer PEC value
i2c_pec_value_get	get packet error checking value
i2c_smbus_alert_config	configure I2C alert through SMBA pin
i2c_smbus_arp_config	configure I2C ARP protocol in SMBus
i2c_sam_enable	enable SAM_V interface
i2c_sam_disable	disable SAM_V interface
i2c_sam_timeout_enable	enable SAM_V interface timeout detect
i2c_sam_timeout_disable	disable SAM_V interface timeout detect
i2c_flag_get	get I2C flag status
i2c_flag_clear	clear I2C flag status
i2c_interrupt_enable	enable I2C interrupt
i2c_interrupt_disable	disable I2C interrupt
i2c_interrupt_flag_get	get I2C interrupt flag status
i2c_interrupt_flag_clear	clear I2C interrupt flag status

Enum i2c_flag_enum

Table 3-258. Enum i2c_flag_enum

Member name	Function description
I2C_FLAG_SBSEND	start condition sent out in master mode
I2C_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode
I2C_FLAG_BTC	byte transmission finishes
I2C_FLAG_ADD10SEND	header of 10-bit address is sent in master mode
I2C_FLAG_STPDET	stop condition detected in slave mode
I2C_FLAG_RBNE	I2C_DATA is not empty during receiving
I2C_FLAG_TBE	I2C_DATA is empty during transmitting
I2C_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus
I2C_FLAG_LOSTARB	arbitration lost in master mode
I2C_FLAG_AERR	acknowledge error
I2C_FLAG_OUERR	over-run or under-run situation occurs in slave mode
I2C_FLAG_PECERR	PEC error when receiving data
I2C_FLAG_SMBTO	timeout signal in SMBus mode
I2C_FLAG_SMBALT	SMBus alert status
I2C_FLAG_MASTER	a flag indicating whether I2C block is in master or slave mode
I2C_FLAG_I2CBSY	busy flag
I2C_FLAG_TR	whether the I2C is a transmitter or a receiver
I2C_FLAG_RXGC	general call address (00h) received
I2C_FLAG_DEFSMB	default address of SMBus device
I2C_FLAG_HSTSMB	SMBus host header detected in slave mode
I2C_FLAG_DUMOD	dual flag in slave mode indicating which address is matched in dual-address mode
I2C_FLAG_TFF	txframe fall flag
I2C_FLAG_TFR	txframe rise flag
I2C_FLAG_RFF	rxframe fall flag
I2C_FLAG_RFR	rxframe rise flag

Enum i2c_interrupt_flag_enum

Table 3-259. Enum i2c_interrupt_flag_enum

Member name	Function description
I2C_INT_FLAG_SBSEND	start condition sent out in master mode interrupt flag
I2C_INT_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode interrupt flag
I2C_INT_FLAG_BTC	byte transmission finishes interrupt flag
I2C_INT_FLAG_ADD10SEND	header of 10-bit address is sent in master mode interrupt flag
I2C_INT_FLAG_STPDET	stop condition detected in slave mode interrupt flag

Member name	Function description
I2C_INT_FLAG_RBNE	I2C_DATA is not empty during receiving interrupt flag
I2C_INT_FLAG_TBE	I2C_DATA is empty during transmitting interrupt flag
I2C_INT_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus interrupt flag
I2C_INT_FLAG_LOSTARB	arbitration lost in master mode interrupt flag
I2C_INT_FLAG_AERR	acknowledge error interrupt flag
I2C_INT_FLAG_OUERR	over-run or under-run situation occurs in slave mode interrupt flag
I2C_INT_FLAG_PECERR	PEC error when receiving data interrupt flag
I2C_INT_FLAG_SMBTO	timeout signal in SMBus mode interrupt flag
I2C_INT_FLAG_SMBALT	SMBus alert status interrupt flag
I2C_INT_FLAG_TFF	txframe fall interrupt flag
I2C_INT_FLAG_TFR	txframe rise interrupt flag
I2C_INT_FLAG_RFF	rxframe fall interrupt flag
I2C_INT_FLAG_RFR	rxframe rise interrupt flag

Enum i2c_interrupt_enum

Table 3-260. Enum i2c_interrupt_enum

Member name	Function description
I2C_INT_ERR	error interrupt
I2C_INT_EV	event interrupt
I2C_INT_BUF	buffer interrupt
I2C_INT_TFF	txframe fall interrupt
I2C_INT_TFR	txframe rise interrupt
I2C_INT_RFF	rxframe fall interrupt
I2C_INT_RFR	rxframe rise interrupt

i2c_deinit

The description of i2c_deinit is shown as below:

Table 3-261. Function i2c_deinit

Function name	i2c_deinit
Function prototype	void i2c_deinit(uint32_t i2c_periph);
Function descriptions	reset I2C
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* reset I2C0 */

i2c_deinit (I2C0);

i2c_clock_config

The description of i2c_clock_config is shown as below:

Table 3-262. Function i2c_clock_config

Function name	i2c_clock_config
Function prototype	void i2c_clock_config(uint32_t i2c_periph, uint32_t clkspeed, uint32_t dutycyc);
Function descriptions	configure I2C clock
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
clkspeed	i2c clock speed
Input parameter{in}	
dutycyc	duty cycle in fast mode
I2C_DTCY_2	T_low/T_high=2
I2C_DTCY_16_9	T_low/T_high=16/9
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure I2C0 clock speed as 100KHz */

i2c_clock_config(I2C0, 100000, I2C_DTCY_2);

i2c_mode_addr_config

The description of i2c_mode_addr_config is shown as below:

Table 3-263. Function i2c_mode_addr_config

Function name	i2c_mode_addr_config
Function prototype	void i2c_mode_addr_config(uint32_t i2c_periph, uint32_t mode, uint32_t addformat, uint32_t addr);

Function descriptions	configure I2C address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
mode	I2C mode select
I2C_I2CMODE_ENABLE	I2C mode
I2C_SMBUSMODE_ENABLE	SMBus mode
Input parameter{in}	
addformat	7bits or 10bits
I2C_ADDFORMAT_7BITS	address format is 7 bits
I2C_ADDFORMAT_10BITS	address format is 10 bits
Input parameter{in}	
addr	I2C address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure I2C0 address as 0x82, using 7 bits */
i2c_mode_addr_config(I2C0, I2C_I2CMODE_ENABLE, I2C_ADDFORMAT_7BITS, 0x82);
```

i2c_smbus_type_config

The description of i2c_smbus_type_config is shown as below:

Table 3-264. Function i2c_smbus_type_config

Function name	i2c_smbus_type_config
Function prototype	void i2c_smbus_type_config(uint32_t i2c_periph, uint32_t type);
Function descriptions	select SMBus type
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	

type	Device or host
I2C_SMBUS_DEVICE	SMBus mode device type
I2C_SMBUS_HOST	SMBus mode host type
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config I2C0 as SMBUS host type*/
i2c_smbus_type_config (I2C0, I2C_SMBUS_HOST);
```

i2c_ack_config

The description of i2c_ack_config is shown as below:

Table 3-265. Function i2c_ack_config

Function name	i2c_ack_config
Function prototype	void i2c_ack_config(uint32_t i2c_periph, uint32_t ack);
Function descriptions	whether or not to send an ACK
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
ack	whether or not to send an ACK
I2C_ACK_ENABLE	ACK will be sent
I2C_ACK_DISABLE	ACK will not be sent
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 will send ACK */
i2c_ack_config (I2C0, I2C_ACK_ENABLE);
```

i2c_ackpos_config

The description of i2c_ackpos_config is shown as below:

Table 3-266. Function i2c_ackpos_config

Function name	i2c_ackpos_config
Function prototype	void i2c_ackpos_config(uint32_t i2c_periph, uint32_t pos);
Function descriptions	configure I2C POAP position
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
pos	ACK position
I2C_ACKPOS_CURRENT	ACKEN bit decides whether or not to send ACK or not for the current byte
I2C_ACKPOS_NEXT	ACKEN bit decides whether or not to send ACK for the next byte
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* The ACK of I2C0 is send for the current frame */

i2c_ackpos_config (I2C0, I2C_ACKPOS_CURRENT);

i2c_master_addressing

The description of i2c_master_addressing is shown as below:

Table 3-267. Function i2c_master_addressing

Function name	i2c_master_addressing
Function prototype	void i2c_master_addressing(uint32_t i2c_periph, uint32_t addr, uint32_t trandirection);
Function descriptions	master sends slave address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
addr	slave address
Input parameter{in}	
trandirection	transmitter or receiver
I2C_TRANSMITTER	transmitter
I2C_RECEIVER	receiver

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* send slave address to I2C bus and I2C0 act as receiver */

i2c_master_addressing(I2C0, 0x82, I2C_RECEIVER);

i2c_dualaddr_enable

The description of i2c_dualaddr_enable is shown as below:

Table 3-268. Function i2c_dualaddr_enable

Function name	i2c_dualaddr_enable
Function prototype	void i2c_dualaddr_enable(uint32_t i2c_periph, uint32_t addr)
Function descriptions	enable dual-address mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
addr	second address in dual-address mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2C0 dual-address*/

i2c_dualaddr_enable (I2C0, 0x80);

i2c_dualaddr_disable

The description of i2c_dualaddr_disable is shown as below:

Table 3-269. Function i2c_dualaddr_disable

Function name	i2c_dualaddr_disable
Function prototype	void i2c_dualaddr_disable(uint32_t i2c_periph)
Function descriptions	disable dual-address mode
Precondition	-
The called functions	-

Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2C0 dual-address */

i2c_dualaddr_disable (I2C0);

i2c_enable

The description of i2c_enable is shown as below:

Table 3-270. Function i2c_enable

Function name	i2c_enable
Function prototype	void i2c_enable(uint32_t i2c_periph);
Function descriptions	enable I2C
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2C0 */

i2c_enable (I2C0);

i2c_disable

The description of i2c_disable is shown as below:

Table 3-271. Function i2c_disable

Function name	i2c_disable
Function prototype	void i2c_disable(uint32_t i2c_periph);
Function descriptions	disable I2C
Precondition	-

The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C0 */

i2c_disable (I2C0);
```

i2c_start_on_bus

The description of i2c_start_on_bus is shown as below:

Table 3-272. Function i2c_start_on_bus

Function name	i2c_start_on_bus
Function prototype	void i2c_start_on_bus(uint32_t i2c_periph);
Function descriptions	generate a START condition on I2C bus
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 send a start condition to I2C bus */

i2c_start_on_bus (I2C0);
```

i2c_stop_on_bus

The description of i2c_stop_on_bus is shown as below:

Table 3-273. Function i2c_stop_on_bus

Function name	i2c_stop_on_bus
Function prototype	void i2c_stop_on_bus(uint32_t i2c_periph);
Function descriptions	generate a STOP condition on I2C bus

Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 generate a STOP condition to I2C bus */
i2c_stop_on_bus (I2C0);
```

i2c_data_transmit

The description of i2c_data_transmit is shown as below:

Table 3-274. Function i2c_data_transmit

Function name	i2c_data_transmit
Function prototype	void i2c_data_transmit(uint32_t i2c_periph, uint8_t data);
Function descriptions	I2C transmit data function
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
data	transmit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 transmit data */
i2c_data_transmit (I2C0, 0x80);
```

i2c_data_receive

The description of i2c_data_receive is shown as below:

Table 3-275. Function i2c_data_receive

Function name	i2c_data_receive
Function prototype	uint8_t i2c_data_receive(uint32_t i2c_periph);
Function descriptions	I2C receive data function
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
uint8_t	0x00..0xFF

Example:

```
/* I2C0 receive data */

uint8_t i2c_receiver;

i2c_receiver = i2c_data_receive(I2C0);
```

i2c_dma_config

The description of i2c_dma_config is shown as below:

Table 3-276. Function i2c_dma_config

Function name	i2c_dma_config
Function prototype	void i2c_dma_config(uint32_t i2c_periph, uint32_t dmastate);
Function descriptions	configure I2C DMA mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
dmastate	on or off
I2C_DMA_ON	enable DMA mode
I2C_DMA_OFF	disable DMA mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 DMA mode enable */

i2c_dma_config (I2C0, I2C_DMA_ON);

i2c_dma_last_transfer_config

The description of i2c_dma_last_transfer_config is shown as below:

Table 3-277. Function i2c_dma_last_transfer_config

Function name	i2c_dma_last_transfer_config
Function prototype	void i2c_dma_last_transfer_config(uint32_t i2c_periph, uint32_t dmalast);
Function descriptions	configure whether next DMA EOT is DMA last transfer or not
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
dmalast	next DMA EOT is the last transfer or not
I2C_DMALST_ON	next DMA EOT is the last transfer
I2C_DMALST_OFF	next DMA EOT is not the last transfer
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* next DMA EOT is the last transfer */

i2c_dma_last_transfer_config (I2C0, I2C_DMALST_ON);

i2c_stretch_scl_low_config

The description of i2c_stretch_scl_low_config is shown as below:

Table 3-278. Function i2c_stretch_scl_low_config

Function name	i2c_stretch_scl_low_config
Function prototype	void i2c_stretch_scl_low_config(uint32_t i2c_periph, uint32_t stretchpara);
Function descriptions	whether to stretch SCL low when data is not ready in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	

stretchpara	SCL stretching enable or disable
I2C_SCLSTRETCH_ENABLE	enable SCL stretching
I2C_SCLSTRETCH_DISABLE	disable SCL stretching
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* stretch SCL low when data is not ready in slave mode */

i2c_stretch_scl_low_config (I2C0, I2C_SCLSTRETCH_ENABLE);

i2c_slave_response_to_gcall_config

The description of i2c_slave_response_to_gcall_config is shown as below:

Table 3-279. Function i2c_slave_response_to_gcall_config

Function name	i2c_slave_response_to_gcall_config
Function prototype	void i2c_slave_response_to_gcall_config(uint32_t i2c_periph, uint32_t gcallpara);
Function descriptions	whether or not to response to a general call
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
gcallpara	response to a general call or not
I2C_GCEN_ENABLE	slave will response to a general call
I2C_GCEN_DISABLE	slave will not response to a general call
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 will response to a general call */

i2c_slave_response_to_gcall_config (I2C0, I2C_GCEN_ENABLE);

i2c_software_reset_config

The description of i2c_software_reset_config is shown as below:

Table 3-280. Function i2c_software_reset_config

Function name	i2c_software_reset_config
Function prototype	void i2c_software_reset_config(uint32_t i2c_periph, uint32_t sreset);
Function descriptions	configure software reset of I2C
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
sreset	reset or not
I2C_SRESET_SET	I2C is under reset
I2C_SRESET_RESET	I2C is not under reset
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* software reset I2C0 */

i2c_software_reset_config (I2C0, I2C_SRESET_SET);
```

i2c_pec_config

The description of i2c_pec_config is shown as below:

Table 3-281. Function i2c_pec_config

Function name	i2c_pec_config
Function prototype	void i2c_pec_config(uint32_t i2c_periph, uint32_t pecstate);
Function descriptions	configure I2C PEC calculation or not
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
pecstate	on or off
I2C_PEC_ENABLE	PEC calculation on
I2C_PEC_DISABLE	PEC calculation off
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* Enable I2C PEC calculation */

i2c_pec_config (I2C0, I2C_PEC_ENABLE);

i2c_pec_transfer_config

The description of i2c_pec_transfer_enable is shown as below:

Table 3-282. Function i2c_pec_transfer_config

Function name	i2c_pec_transfer_config
Function prototype	void i2c_pec_transfer_config(uint32_t i2c_periph, uint32_t pecpara);
Function descriptions	configure whether to transfer PEC value
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
pecpara	Transfer PEC or not
I2C_PECTRANS_ENABLE	transfer PEC
I2C_PECTRANS_DISABLE	not transfer PEC
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* I2C0 transfer PEC */

i2c_pec_transfer_config (I2C0, I2C_PECTRANS_ENABLE);

i2c_pec_value_get

The description of i2c_pec_value_get is shown as below:

Table 3-283. Function i2c_pec_value_get

Function name	i2c_pec_value_get
Function prototype	uint8_t i2c_pec_value_get(uint32_t i2c_periph);

Function descriptions	get packet error checking value
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
uint8_t	PEC value

Example:

```
/* I2C0 get packet error checking value */
uint8_t pec_value;
pec_value = i2c_pec_value_get (I2C0);
```

i2c_smbus_alert_config

The description of i2c_smbus_alert_config is shown as below:

Table 3-284. Function i2c_smbus_alert_config

Function name	i2c_smbus_alert_config
Function prototype	void i2c_smbus_alert_config (uint32_t i2c_periph, uint32_t smbuspara);
Function descriptions	configure I2C alert through SMBA pin
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
smbuspara	issue alert through SMBA pin or not
I2C_SALTSEND_ENABLE	issue alert through SMBA pin
I2C_SALTSEND_DISABLE	not issue alert through SMBA pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 issue alert through SMBA pin enable*/
```

i2c_smbus_alert_config (I2C0, I2C_SALTSEND_ENABLE);

i2c_smbus_arp_config

The description of i2c_smbus_arp_config is shown as below:

Table 3-285. Function i2c_smbus_arp_config

Function name	i2c_smbus_arp_config
Function prototype	void i2c_smbus_arp_config(uint32_t i2c_periph, uint32_t arpstate);
Function descriptions	configure I2C ARP protocol in SMBus
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
arpstate	ARP protocol in SMBus switch
I2C_ARP_ENABLE	enable ARP
I2C_ARP_DISABLE	disable ARP
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2C0 ARP protocol in SMBus switch */

i2c_smbus_arp_config (I2C0, I2C_ARP_ENABLE);

i2c_sam_enable

The description of i2c_sam_enable is shown as below:

Table 3-286. Function i2c_sam_enable

Function name	i2c_sam_enable
Function prototype	void i2c_sam_enable(uint32_t i2c_periph);
Function descriptions	enable SAM_V interface
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable I2C0 SAM_V interface */

i2c_sam_enable (I2C0);

i2c_sam_disable

The description of i2c_sam_disable is shown as below:

Table 3-287. Function i2c_sam_disable

Function name	i2c_sam_disable
Function prototype	void i2c_sam_disable(uint32_t i2c_periph);
Function descriptions	disable SAM_V interface
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2C0 SAM_V interface */

i2c_sam_disable (I2C0);

i2c_sam_timeout_enable

The description of i2c_sam_timeout_enable is shown as below:

Table 3-288. Function i2c_sam_timeout_enable

Function name	i2c_sam_timeout_enable
Function prototype	void i2c_sam_timeout_enable(uint32_t i2c_periph);
Function descriptions	enable SAM_V interface timeout detect
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* enable I2C0 SAM_V interface timeout detect */

i2c_sam_timeout_enable (I2C0);

i2c_sam_timeout_disable

The description of i2c_sam_timeout_disable is shown as below:

Table 3-289. Function i2c_sam_timeout_disable

Function name	i2c_sam_timeout_disable
Function prototype	void i2c_sam_timeout_disable(uint32_t i2c_periph);
Function descriptions	disable SAM_V interface timeout detect
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2C0 SAM_V interface timeout detect */

i2c_sam_timeout_disable (I2C0);

i2c_flag_get

The description of i2c_flag_get is shown as below:

Table 3-290. Function i2c_flag_get

Function name	i2c_flag_get
Function prototype	FlagStatus i2c_flag_get(uint32_t i2c_periph, i2c_flag_enum flag)
Function descriptions	check whether the I2C flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	

flag	specify get which flag
I2C_FLAG_SBSEND	start condition sent out in master mode
I2C_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode
I2C_FLAG_BTC	byte transmission finishes
I2C_FLAG_ADD10SEND	header of 10-bit address is sent in master mode
I2C_FLAG_STPDET	stop condition detected in slave mode
I2C_FLAG_RBNE	I2C_DATA is not empty during receiving
I2C_FLAG_TBE	I2C_DATA is empty during transmitting
I2C_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus
I2C_FLAG_LOSTARB	arbitration lost in master mode
I2C_FLAG_AERR	acknowledge error
I2C_FLAG_OUERR	over-run or under-run situation occurs in slave mode
I2C_FLAG_PECERR	PEC error when receiving data
I2C_FLAG_SMBTO	timeout signal in SMBus mode
I2C_FLAG_SMBALT	SMBus alert status
I2C_FLAG_MASTER	a flag indicating whether I2C block is in master or slave mode
I2C_FLAG_I2CBSY	busy flag
I2C_FLAG_TR	whether the I2C is a transmitter or a receiver
I2C_FLAG_RXGC	general call address (00h) received
I2C_FLAG_DEFSMB	default address of SMBus device
I2C_FLAG_HSTSMB	SMBus host header detected in slave mode
I2C_FLAG_DUMOD	dual flag in slave mode indicating which address is matched in dual-address mode
I2C_FLAG_TFF	txframe fall flag
I2C_FLAG_TFR	txframe rise flag
I2C_FLAG_RFF	rxframe fall flag
I2C_FLAG_RFR	rxframe rise flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* check whether start condition send out */
FlagStatus flag_state = RESET;
flag_state = i2c_flag_get (I2C0, I2C_FLAG_SBSEND);
```

i2c_flag_clear

The description of i2c_flag_clear is shown as below:

Table 3-291. Function i2c_flag_clear

Function name	i2c_flag_clear
Function prototype	void i2c_flag_clear(uint32_t i2c_periph, i2c_flag_enum flag)
Function descriptions	clear I2C flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
flag	flag type
I2C_FLAG_SMBALT	SMBus Alert status
I2C_FLAG_SMBTO	timeout signal in SMBus mode
I2C_FLAG_PECERR	PEC error when receiving data
I2C_FLAG_OUERR	over-run or under-run situation occurs in slave mode
I2C_FLAG_AERR	acknowledge error
I2C_FLAG_LOSTARB	arbitration lost in master mode
I2C_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus
I2C_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode
I2C_FLAG_TFF	txframe fall flag
I2C_FLAG_TFR	txframe rise flag
I2C_FLAG_RFF	rxframe fall flag
I2C_FLAG_RFR	rxframe rise flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear a bus error flag*/
i2c_flag_clear (I2C0, I2C_FLAG_BERR);
```

i2c_interrupt_enable

The description of i2c_interrupt_enable is shown as below:

Table 3-292. Function i2c_interrupt_enable

Function name	i2c_interrupt_enable
Function prototype	void i2c_interrupt_enable(uint32_t i2c_periph, i2c_interrupt_enum interrupt);
Function descriptions	enable I2C interrupt
Precondition	-
The called functions	-

Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
interrupt	interrupt type
I2C_INT_ERR	error interrupt
I2C_INT_EV	event interrupt
I2C_INT_BUF	buffer interrupt
I2C_INT_TFF	txframe fall interrupt
I2C_INT_TFR	txframe rise interrupt
I2C_INT_RFF	rxframe fall interrupt
I2C_INT_RFR	rxframe rise interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C0 event interrupt */

i2c_interrupt_enable (I2C0, I2C_INT_EV);
```

i2c_interrupt_disable

The description of i2c_interrupt_disable is shown as below:

Table 3-293. Function i2c_interrupt_disable

Function name	i2c_interrupt_disable
Function prototype	void i2c_interrupt_disable(uint32_t i2c_periph, i2c_interrupt_enum interrupt);
Function descriptions	disable I2C interrupt
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
inttype	interrupt type
I2C_INT_ERR	error interrupt
I2C_INT_EV	event interrupt
I2C_INT_BUF	buffer interrupt
I2C_INT_TFF	txframe fall interrupt
I2C_INT_TFR	txframe rise interrupt
I2C_INT_RFF	rxframe fall interrupt

I2C_INT_RFR	rxframe rise interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2C0 event interrupt */

i2c_interrupt_disable (I2C0, I2C_INT_EV);

i2c_interrupt_flag_get

The description of i2c_interrupt_flag_get is shown as below:

Table 3-294. Function i2c_interrupt_flag_get

Function name	i2c_interrupt_flag_get
Function prototype	FlagStatus i2c_interrupt_flag_get(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag)
Function descriptions	get I2C interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
int_flag	interrupt flag
I2C_INT_FLAG_SBSEND	start condition sent out in master mode interrupt flag
I2C_INT_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode interrupt flag
I2C_INT_FLAG_BTC	byte transmission finishes
I2C_INT_FLAG_ADD10SEND	header of 10-bit address is sent in master mode interrupt flag
I2C_INT_FLAG_STPDET	stop condition detected in slave mode interrupt flag
I2C_INT_FLAG_RBNE	I2C_DATA is not Empty during receiving interrupt flag
I2C_INT_FLAG_TBE	I2C_DATA is empty during transmitting interrupt flag
I2C_INT_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus interrupt flag
I2C_INT_FLAG_LOSTARB	arbitration lost in master mode interrupt flag
I2C_INT_FLAG_AERR	acknowledge error interrupt flag
I2C_INT_FLAG_OUER	over-run or under-run situation occurs in slave mode interrupt flag

R	
I2C_INT_FLAG_PECERR	PEC error when receiving data interrupt flag
I2C_INT_FLAG_SMBTO	timeout signal in SMBus mode interrupt flag
I2C_INT_FLAG_SMBALT	SMBus alert status interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* check the byte transmission finishes interrupt flag is set or not*/
FlagStatus flag_state = RESET;
flag_state = i2c_interrupt_flag_get (I2C0, I2C_INT_FLAG_BTC);
```

i2c_interrupt_flag_clear

The description of i2c_interrupt_flag_clear is shown as below:

Table 3-295. Function i2c_interrupt_flag_clear

Function name	i2c_interrupt_flag_clear
Function prototype	void i2c_interrupt_flag_clear(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag);
Function descriptions	clear I2C interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
int_flag	interrupt flag
I2C_INT_FLAG_ADDSEND	address is sent in master mode or received and matches in slave mode interrupt flag
I2C_INT_FLAG_BERR	a bus error occurs indication a unexpected start or stop condition on I2C bus interrupt flag
I2C_INT_FLAG_LOSTARB	arbitration lost in master mode interrupt flag
I2C_INT_FLAG_AERR	acknowledge error interrupt flag
I2C_INT_FLAG_OUERR	over-run or under-run situation occurs in slave mode interrupt flag
I2C_INT_FLAG_PECE	PEC error when receiving data interrupt flag

RR	
I2C_INT_FLAG_SMBTO	timeout signal in SMBus mode interrupt flag
I2C_INT_FLAG_SMBALT	SMBus alert status interrupt flag
I2C_INT_FLAG_TFF	txframe fall interrupt flag
I2C_INT_FLAG_TFR	txframe rise interrupt flag
I2C_INT_FLAG_RFF	rxframe fall interrupt flag
I2C_INT_FLAG_RFR	rxframe rise interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the acknowledge error interrupt flag */
i2c_interrupt_flag_clear (I2C0, I2C_INT_FLAG_AERR);
```

3.15. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick.The NVIC and SysTick registers are listed in chapter *3.15.1*, the MISC firmware functions are introduced in chapter *3.15.2*.

3.15.1. Descriptions of Peripheral registers

Table 3-296. NVIC Registers

Registers	Descriptions
ISER(1)	Interrupt Set Enable Register
ICER(1)	Interrupt Clear Enable Register
ISPR(1)	Interrupt Set Pending Register
ICPR(1)	Interrupt Clear Pending Register
IABR(1)	Interrupt Active bit Register
IP(1)	Interrupt Priority Register
STIR(1)	Software Trigger Interrupt Register
CPUID(2)	CPUID Base Register
ICSR(2)	Interrupt Control and State Register
VTOR(2)	Vector Table Offset Register
AIRCR(2)	Application Interrupt and Reset Control Register
SCR(2)	System Control Register
CCR(2)	Configuration Control Register

Registers	Descriptions
SHP(2)	System Handlers Priority Registers
SHCSR(2)	System Handler Control and State Register
CFSR(2)	Configurable Fault Status Register
HFSR(2)	HardFault Status Register
DFSR(2)	Debug Fault Status Register
MMFAR(2)	MemManage Fault Address Register
BFAR(2)	BusFault Address Register
AFSR(2)	Auxiliary Fault Status Register
PFR(2)	Processor Feature Register
DFR(2)	Debug Feature Register
ADR(2)	Auxiliary Feature Register
MMFR(2)	Memory Model Feature Register
ISAR(2)	Instruction Set Attributes Register
CPACR(2)	Coprocessor Access Control Register

1. refer to the structure NVIC_Type, is defined in the core_cm4h file
2. refer to the structure SCB_Type, is defined in the core_cm4. file

Table 3-297. SysTick Registers

Registers	Descriptions
CTRL(1)	SysTick Control and Status Register
LOAD(1)	SysTick Reload Value Register
VAL(1)	SysTick Current Value Register
CALIB(1)	SysTick Calibration Register

1. refer to the structure SysTick_Type, is defined in the core_cm4h file

3.15.2. Descriptions of Peripheral functions

Enum IRQn_Type

Table 3-298. IRQn_Type

Member name	Function description
WWDGT_IRQn	WWDGT interrupt
LVD_IRQn	LVD from EXTI interrupt
TAMPER_IRQn	Tamper interrupt
RTC_IRQn	RTC global interrupt
FMC_IRQn	FMC global interrupt
RCU_CTC_IRQn	RCU and CTC interrupts
EXTI0_IRQn	EXTI line0 interrupt
EXTI1_IRQn	EXTI line1 interrupt
EXTI2_IRQn	EXTI line2 interrupt
EXTI3_IRQn	EXTI line3 interrupt
EXTI4_IRQn	EXTI line4 interrupt

Member name	Function description
DMA0_Channel0_IRQn	DMA0 channel0 global interrupt
DMA0_Channel1_IRQn	DMA0 channel1 global interrupt
DMA0_Channel2_IRQn	DMA0 channel2 global interrupt
DMA0_Channel3_IRQn	DMA0 channel3 global interrupt
DMA0_Channel4_IRQn	DMA0 channel4 global interrupt
DMA0_Channel5_IRQn	DMA0 channel5 global interrupt
DMA0_Channe6_IRQn	DMA0 channel6 global interrupt
ADC0_1_IRQn	ADC0 and ADC1 interrupts
EXTI5_9_IRQn	EXTI line[9:5] interrupts
TIMER0_BRK_TIMER8_IRQn	TIMER0 break and TIMER8 global interrupts
TIMER0_UP_TIMER9_IRQn	TIMER0 update and TIMER9 global interrupts
TIMER0_TRG_CMT_TIMER10_IRQn	TIMER0 trigger and channel commutation and TIMER10 interrupts
TIMER0_Channel_IRQn	TIMER0 channel capture compare interrupt
TIMER1_IRQn	TIMER1 global interrupt
TIMER2_IRQn	TIMER2 global interrupt
TIMER3_IRQn	TIMER3 global interrupt
I2C0_EV_IRQn	I2C0 event interrupt
I2C0_ER_IRQn	I2C0 error interrupt
I2C1_EV_IRQn	I2C1 event interrupt
I2C1_ER_IRQn	I2C1 error interrupt
SPI0_IRQn	SPI0 global interrupt
SPI1_IRQn	SPI1 global interrupt
USART0_IRQn	USART0 global interrupt
USART1_IRQn	USART1 global interrupt
USART2_IRQn	USART2 global interrupt
EXTI10_15_IRQn	EXTI line[15:10] interrupts
RTC_Alarm_IRQn	RTC alarm from EXTI interrupt
USBFS_WKUP_IRQn	USBFS wakeup from EXTI interrupt
TIMER7_BRK_TIMER11_IRQn	TIMER7 break and TIMER11 global interrupts
TIMER7_UP_TIMER12_IRQn	TIMER7 update and TIMER12 global interrupts
TIMER7_TRG_CMT_TIMER13_IRQn	TIMER7 trigger and Channel commutation and TIMER13 global interrupts
TIMER7_Channel_IRQn	TIMER7 channel capture compare interrupt
EXMC_IRQn	EXMC global interrupt
TIMER4_IRQn	TIMER4 global interrupt
SPI2_IRQn	SPI2 global interrupt
UART3_IRQn	UART3 global interrupt
UART4_IRQn	UART4 global interrupt
TIMER5_IRQn	TIMER5 global interrupt
TIMER6_IRQn	TIMER6 global interrupt

Member name	Function description
DMA1_Channel0_IRQn	DMA1 channel0 global interrupt
DMA1_Channel1_IRQn	DMA1 channel1 global interrupt
DMA1_Channel2_IRQn	DMA1 channel2 global interrupt
DMA1_Channel3_IRQn	DMA1 channel3 global interrupt
DMA1_Channel4_IRQn	DMA1 channel4 global interrupt
USBFS_IRQn	USBFS global interrupt

MISC firmware functions are listed in the table shown as below:

Table 3-299. MISC firmware function

Function name	Function description
nvic_priority_group_set	set the priority group
nvic_irq_enable	enable NVIC interrupt request
nvic_irq_disable	disable NVIC interrupt request
nvic_vector_table_set	set the NVIC vector table base address
system_lowpower_set	set the state of the low power mode
system_lowpower_reset	reset the state of the low power mode
systick_clksource_set	set the systick clock source

nvic_priority_group_set

The description of nvic_priority_group_set is shown as below:

Table 3-300. Function nvic_priority_group_set

Function name	nvic_priority_group_set
Function prototype	void nvic_priority_group_set(uint32_t nvic_prigroup);
Function descriptions	set the priority group
Precondition	-
The called functions	-
Input parameter{in}	
nvic_prigroup	priority group
NVIC_PRIGROUP_PRE0_SUB4	0 bits for pre-emption priority, 4 bits for subpriority
NVIC_PRIGROUP_PRE1_SUB3	1 bits for pre-emption priority, 3 bits for subpriority
NVIC_PRIGROUP_PRE2_SUB2	2 bits for pre-emption priority, 2 bits for subpriority
NVIC_PRIGROUP_PRE3_SUB1	3 bits for pre-emption priority, 1 bits for subpriority
NVIC_PRIGROUP_PRE4_SUB0	4 bits for pre-emption priority, 0 bits for subpriority
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* priority group configuration , 0 bits for pre-emption priority, 4 bits for subpriority */

nvic_priority_group_set(NVIC_PRIGROUP_PRE0_SUB4);

nvic_irq_enable

The description of nvic_irq_enable is shown as below:

Table 3-301. Function nvic_irq_enable

Function name	nvic_irq_enable
Function prototype	void nvic_irq_enable(uint8_t nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t nvic_irq_sub_priority);
Function descriptions	enable NVIC interrupt request
Precondition	-
The called functions	nvic_priority_group_set
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to enum *Table 3-298. IRQn_Type*
Input parameter{in}	
nvic_irq_pre_priority	the pre-emption priority needed to set
Input parameter{in}	
nvic_irq_sub_priority	the subpriority needed to set
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable window watchDog timer interrupt, pre-emption priority is 1, subpriority is 1 */
nvic_irq_enable(WWDGT_IRQn, 1, 1);

nvic_irq_disable

The description of nvic_irq_disable is shown as below:

Table 3-302. Function nvic_irq_disable

Function name	nvic_irq_disable
Function prototype	void nvic_irq_disable(uint8_t nvic_irq);
Function descriptions	disable NVIC interrupt request
Precondition	-
The called functions	-
Input parameter{in}	

nvic_irq	NVIC interrupt, refer to enum *Table 3-298. IRQn_Type*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable window watchDog timer interrupt */
nvic_irq_disable(WWDGT_IRQn);
```

nvic_vector_table_set

The description of nvic_vector_table_set is shown as below:

Table 3-303. Function nvic_vector_table_set

Function name	nvic_vector_table_set
Function prototype	void nvic_vector_table_set(uint32_t nvic_vict_tab, uint32_t offset);
Function descriptions	set the NVIC vector table base address
Precondition	-
The called functions	-
Input parameter{in}	
nvic_vict_tab	the RAM or FLASH base address
NVIC_VECTTAB_RAM	RAM base address
NVIC_VECTTAB_FLASH	Flash base address
Input parameter{in}	
offset	vector table offset (vector table start address= base address+offset)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set vector table address = NVIC_VECTTAB_FLASH +0x200 */
nvic_vector_table_set (NVIC_VECTTAB_FLASH,0x200);
```

system_lowpower_set

The description of system_lowpower_set is shown as below:

Table 3-304. Function system_lowpower_set

Function name	system_lowpower_set
Function prototype	void system_lowpower_set(uint8_t lowpower_mode);
Function descriptions	set the state of the low power mode

Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state
SCB_LPM_SLEEP_EXIT_ISR	if chose this para, the system always enter low power mode by exiting from ISR
SCB_LPM_DEEPSLEEP	if chose this para, the system will enter the DEEPSLEEP mode
SCB_LPM_WAKE_BY_ALL_INT	if chose this para, the lowpower mode can be woke up by all the enable and disable interrupts
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the system always enter low power mode by exiting from ISR */
system_lowpower_set (SCB_LPM_SLEEP_EXIT_ISR);
```

system_lowpower_reset

The description of system_lowpower_reset is shown as below:

Table 3-305. Function system_lowpower_reset

Function name	system_lowpower_reset
Function prototype	void system_lowpower_reset(uint8_t lowpower_mode);
Function descriptions	reset the state of the low power mode
Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state
SCB_LPM_SLEEP_EXIT_ISR	if chose this para, the system will exit low power mode by exiting from ISR
SCB_LPM_DEEPSLEEP	if chose this para, the system will enter the SLEEP mode
SCB_LPM_WAKE_BY_ALL_INT	if chose this para, the lowpower mode only can be woke up by the enable interrupts
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* the system will exit low power mode by exiting from ISR */
```

system_lowpower_reset (SCB_LPM_SLEEP_EXIT_ISR);

systick_clksource_set

The description of systick_clksource_set is shown as below:

Table 3-306. Function systick_clksource_set

Function name	systick_clksource_set
Function prototype	void systick_clksource_set(uint32_t systick_clksource);
Function descriptions	set the systick clock source
Precondition	-
The called functions	-
Input parameter{in}	
systick_clksource	the systick clock source needed to choose
SYSTICK_CLKSOURCE_HCLK	systick clock source is from HCLK
SYSTICK_CLKSOURCE_HCLK_DIV8	systick clock source is from HCLK/8
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* systick clock source is HCLK/8 */
systick_clksource_set (SYSTICK_CLKSOURCE_HCLK_DIV8);
```

3.16. PMU

According to the Power management unit (PMU), provides three types of power saving modes, including Sleep, Deep-sleep and Standby mode. The PMU registers are listed in chapter *3.16.1*, the PMU firmware functions are introduced in chapter *3.16.2*.

3.16.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

Table 3-307. PMU Registers

Registers	Descriptions
PMU_CTL	Control register
PMU_CS	Control and status register

3.16.2. Descriptions of Peripheral functions

PMU firmware functions are listed in the table shown as below:

Table 3-308. PMU firmware function

Function name	Function description
pmu_deinit	deinitialize the PMU
pmu_lvd_select	select low voltage detector threshold
pmu_ldo_output_select	select LDO output voltage
pmu_lvd_disable	disable PMU lvd
pmu_to_sleepmode	PMU work at sleep mode
pmu_to_deepsleepmode	PMU work at deepsleep mode
pmu_to_standbymode	pmu work at standby mode
pmu_wakeup_pin_enable	enable wakeup pin
pmu_wakeup_pin_disable	disable wakeup pin
pmu_backup_write_enable	enable backup domain write
pmu_backup_write_disable	disable backup domain write
pmu_flag_get	get flag state
pmu_flag_clear	clear flag bit

pmu_deinit

The description of pmu_deinit is shown as below:

Table 3-309. Function pmu_deinit

Function name	pmu_deinit
Function prototype	void pmu_deinit(void);
Function descriptions	deinitialize the PMU
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset PMU */

pmu_deinit();
```

pmu_lvd_select

The description of pmu_lvd_select is shown as below:

Table 3-310. Function pmu_lvd_select

Function name	pmu_lvd_select
Function prototype	void pmu_lvd_select(uint32_t lvdt_n);
Function descriptions	select low voltage detector threshold
Precondition	-
The called functions	-
Input parameter{in}	
lvdt_n	voltage threshold value
PMU_LVDT_0	voltage threshold is 2.2V
PMU_LVDT_1	voltage threshold is 2.3V
PMU_LVDT_2	voltage threshold is 2.4V
PMU_LVDT_3	voltage threshold is 2.5V
PMU_LVDT_4	voltage threshold is 2.6V
PMU_LVDT_5	voltage threshold is 2.7V
PMU_LVDT_6	voltage threshold is 2.8V
PMU_LVDT_7	voltage threshold is 2.9V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select low voltage detector threshold as 2.9V */
pmu_lvd_select(PMU_LVDT_7);
```

pmu_ldo_output_select

The description of pmu_ldo_output_select is shown as below:

Table 3-311. Function pmu_ldo_output_select

Function name	pmu_ldo_output_select
Function prototype	void pmu_ldo_output_select(uint32_t ldo_output);
Function descriptions	internal voltage regulator (LDO) output voltage select
Precondition	-
The called functions	-
Input parameter{in}	
ldo_output	output voltage mode
PMU_LDOVS_LOW	output low voltage mode
PMU_LDOVS_NORMAL	output normal voltage mode
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* select output low voltage mode */

pmu_ldo_output_select(PMU_LDOVS_LOW);

pmu_lvd_disable

The description of pmu_lvd_disable is shown as below:

Table 3-312. Function pmu_lvd_disable

Function name	pmu_lvd_disable
Function prototype	void pmu_lvd_disable(void);
Function descriptions	disable PMU lvd
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable PMU lvd */

pmu_lvd_disable();

pmu_to_sleepmode

The description of pmu_to_sleepmode is shown as below:

Table 3-313. Function pmu_to_sleepmode

Function name	pmu_to_sleepmode
Function prototype	void pmu_to_sleepmode(uint8_t sleepmodecmd);
Function descriptions	PMU work at sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
sleepmodecmd	command to enter sleep mode
WFI_CMD	use WFI command
WFE_CMD	use WFE command
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* PMU work at sleep mode */

pmu_to_sleepmode(WFI_CMD);

pmu_to_deepsleepmode

The description of pmu_to_deepsleepmode is shown as below:

Table 3-314. Function pmu_to_deepsleepmode

Function name	pmu_to_deepsleepmode
Function prototype	void pmu_to_deepsleepmode(uint32_t ldo,uint8_t deepsleepmodecmd);
Function descriptions	PMU work at deepsleep mode
Precondition	-
The called functions	-
Input parameter{in}	
ldo	ldo work mode
PMU_LDO_NORMAL	LDO normal work when pmu enter deepsleep mode
PMU_LDO_LOWPOWER	LDO work at low power mode when pmu enter deepsleep mode
Input parameter{in}	
deepsleepmodecmd	command to enter deepsleep mode
WFI_CMD	use WFI command
WFE_CMD	use WFE command
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* PMU work at deepsleep mode */

pmu_to_deepsleepmode(PMU_LDO_NORMAL, WFI_CMD);

pmu_to_standbymode

The description of pmu_to_standbymode is shown as below:

Table 3-315. Function pmu_to_standbymode

Function name	pmu_to_standbymode
Function prototype	void pmu_to_standbymode(void);
Function descriptions	pmu work at standby mode

Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PMU work at standby mode */
pmu_to_standbymode();
```

pmu_wakeup_pin_enable

The description of pmu_wakeup_pin_enable is shown as below:

Table 3-316. Function pmu_wakeup_pin_enable

Function name	pmu_wakeup_pin_enable
Function prototype	void pmu_wakeup_pin_enable(void);
Function descriptions	enable wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable wakeup pin */
pmu_wakeup_pin_enable();
```

pmu_wakeup_pin_disable

The description of pmu_wakeup_pin_disable is shown as below:

Table 3-317. Function pmu_wakeup_pin_disable

Function name	pmu_wakeup_pin_disable
Function prototype	void pmu_wakeup_pin_disable(void);
Function descriptions	disable wakeup pin
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable wakeup pin */

pmu_wakeup_pin_disable();
```

pmu_backup_write_enable

The description of pmu_backup_write_enable is shown as below:

Table 3-318. Function pmu_backup_write_enable

Function name	pmu_backup_write_enable
Function prototype	void pmu_backup_write_enable(void);
Function descriptions	enable backup domain write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable backup domain write */

pmu_backup_write_enable();
```

pmu_backup_write_disable

The description of pmu_backup_write_disable is shown as below:

Table 3-319. Function pmu_backup_write_disable

Function name	pmu_backup_write_disable
Function prototype	void pmu_backup_write_disable(void);
Function descriptions	disable backup domain write
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable backup domain write */
pmu_backup_write_disable();
```

pmu_flag_get

The description of pmu_flag_get is shown as below:

Table 3-320. Function pmu_flag_get

Function name	pmu_flag_get
Function prototype	FlagStatus pmu_flag_get(uint32_t flag);
Function descriptions	get flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag
PMU_FLAG_WAKEUP	wakeup flag
PMU_FLAG_STANDBY	standby flag
PMU_FLAG_LVD	low voltage detector status flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get flag state */
FlagStatus status;
status = pmu_flag_get(PMU_FLAG_WAKEUP);
```

pmu_flag_clear

The description of pmu_flag_clear is shown as below:

Table 3-321. Function pmu_flag_clear

Function name	pmu_flag_clear
Function prototype	void pmu_flag_clear(uint32_t flag_reset);

Function descriptions	clear flag bit
Precondition	-
The called functions	-
Input parameter{in}	
flag_reset	flag
PMU_FLAG_RESET_WAKEUP	reset wakeup flag
PMU_FLAG_RESET_STANDBY	reset standby flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear flag bit */
pmu_flag_clear(PMU_FLAG_RESET_WAKEUP);
```

3.17. RCU

RCU is the reset and clock unit. Reset Control includes the control of three kinds of reset: power reset, system reset and backup domain reset. The Clock Control unit provides a range of frequencies and clock functions. The RCU registers are listed in chapter *3.17.1*, the RCU firmware functions are introduced in chapter *3.17.2*.

3.17.1. Descriptions of Peripheral registers

Table 3-322. RCU Registers

Registers	Descriptions
RCU_CTL	Control register
RCU_CFG0	Clock configuration register 0
RCU_INT	Clock interrupt register
RCU_APB2RST	APB2 reset register
RCU_APB1RST	APB1 reset register
RCU_AHBEN	AHB enable register
RCU_APB2EN	APB2 enable register
RCU_APB1EN	APB1 enable register
RCU_BDCTL	Backup domain control register
RCU_RSTSCK	Reset source/clock register
RCU_AHBRST	AHB reset register
RCU_CFG1	Clock configuration register 1

Registers	Descriptions
RCU_DSV	Deep-sleep mode voltage register
RCU_ADDCTL	Additional clock control register
RCU_ADDINT	Additional clock interrupt register
RCU_ADDAPB1RST	APB1 additional reset register
RCU_ADDAPB1EN	APB1 additional enable register

3.17.2. Descriptions of Peripheral functions

RCU firmware function are listed in the table shown as below:

Table 3-323. RCU firmware function

Function name	Function description
rcu_deinit	deinitialize the RCU
rcu_periph_clock_enable	enable the peripherals clock
rcu_periph_clock_disable	disable the peripherals clock
rcu_periph_clock_sleep_enable	enable the peripherals clock when in sleep mode
rcu_periph_clock_sleep_disable	disable the peripherals clock when in sleep mode
rcu_periph_reset_enable	enable the peripherals reset
rcu_periph_reset_disable	disable the peripheral reset
rcu_bkp_reset_enable	enable the BKP domain reset
rcu_bkp_reset_disable	disable the BKP domain reset
rcu_system_clock_source_config	configure the system clock source
rcu_system_clock_source_get	get the system clock source
rcu_ahb_clock_config	configure the AHB clock prescaler selection
rcu_apb1_clock_config	configure the APB1 clock prescaler selection
rcu_apb2_clock_config	configure the APB2 clock prescaler selection
rcu_ckout0_config	configure the CK_OUT0 clock source
rcu_pll_config	configure the main PLL clock
rcu_pllpresel_config	configure the PLL clock source preselection
rcu_predv0_config	configure the PREDV0 division factor
rcu_predv1_config	configure the PREDV1 division factor
rcu_pll1_config	configure the PLL1 clock
rcu_pll2_config	configure the PLL2 clock
rcu_adc_clock_config	configure the ADC prescaler factor
rcu_usb_clock_config	configure the USB prescaler factor
rcu_rtc_clock_config	configure the RTC clock source selection
rcu_i2s1_clock_config	configure the I2S1 clock source selection
rcu_i2s2_clock_config	configure the I2S2 clock source selection
rcu_ck48m_clock_config	configure the CK48M clock selection
rcu_flag_get	get the clock stabilization and periphral reset flags
rcu_all_reset_flag_clear	clear all the reset flag
rcu_interrupt_flag_get	get the clock stabilization interrupt and ckm flags

Function name	Function description
rcu_interrupt_flag_clear	clear the interrupt flags
rcu_interrupt_enable	enable the stabilization interrupt
rcu_interrupt_disable	disable the stabilization interrupt
rcu_lxtal_drive_capability_config	configure the LXTAL drive capability
rcu_osci_stab_wait	wait for oscillator stabilization flags is SET or oscillator startup is timeout
rcu_osci_on	turn on the oscillator
rcu_osci_off	turn off the oscillator
rcu_osci_bypass_mode_enable	enable the oscillator bypass mode
rcu_osci_bypass_mode_disable	disable the oscillator bypass mode
rcu_hxtal_clock_monitor_enable	enable the HXTAL clock monitor
rcu_hxtal_clock_monitor_disable	disable the HXTAL clock monitor
rcu_irc8m_adjust_value_set	set the IRC8M adjust value
rcu_deepsleep_voltage_set	set the deep-sleep mode voltage value
rcu_clock_freq_get	get the system clock, bus clock frequency

Enum rcu_periph_enum

Table 3-324. rcu_periph_enum

enum name	Function description
RCU_DMA0	DMA0 clock
RCU_DMA1	DMA1 clock
RCU_CRC	CRC clock
RCU_EXMC	EXMC clock
RCU_USBFS	USBFS clock
RCU_TIMER1	TIMER1 clock
RCU_TIMER2	TIMER2 clock
RCU_TIMER3	TIMER3 clock
RCU_TIMER4	TIMER4 clock
RCU_TIMER5	TIMER5 clock
RCU_TIMER6	TIMER6 clock
RCU_TIMER11	TIMER11 clock
RCU_TIMER12	TIMER12 clock
RCU_TIMER13	TIMER13 clock
RCU_WWDGT	WWDGT clock
RCU_SPI1	SPI1 clock
RCU_SPI2	SPI2 clock
RCU_USART1	USART1 clock
RCU_USART2	USART2 clock
RCU_UART3	UART3 clock
RCU_UART4	UART4 clock
RCU_I2C0	I2C0 clock

enum name	Function description
RCU_I2C1	I2C1 clock
RCU_BKPI	BKPI clock
RCU_PMU	PMU clock
RCU_DAC	DAC clock
RCU_RTC	RTC clock
RCU_CTC	CTC clock
RCU_AF	alternate function clock
RCU_GPIOA	GPIOA clock
RCU_GPIOB	GPIOB clock
RCU_GPIOC	GPIOC clock
RCU_GPIOD	GPIOD clock
RCU_GPIOE	GPIOE clock
RCU_ADC0	ADC0 clock
RCU_ADC1	ADC1 clock
RCU_TIMER0	TIMER0 clock
RCU_SPI0	SPI0 clock
RCU_TIMER7	TIMER7 clock
RCU_USART0	USART0 clock
RCU_ADC2	ADC2 clock
RCU_TIMER8	TIMER8 clock
RCU_TIMER9	TIMER9 clock
RCU_TIMER10	TIMER10 clock

Enum rcu_periph_sleep_enum

Table 3-325. rcu_periph_sleep_enum

enum name	Function description
RCU_SRAM_SLP	SRAM clock when sleep mode
RCU_FMC_SLP	FMC clock when sleep mode

Enum rcu_periph_reset_enum

Table 3-326. rcu_periph_reset_enum

enum name	Function description
RCU_USBFSRST	USBFS clock reset
RCU_TIMER1RST	TIMER1 clock reset
RCU_TIMER2RST	TIMER2 clock reset
RCU_TIMER3RST	TIMER3 clock reset
RCU_TIMER4RST	TIMER4 clock reset
RCU_TIMER5RST	TIMER5 clock reset
RCU_TIMER6RST	TIMER6 clock reset
RCU_TIMER11RST	TIMER11 clock reset

enum name	Function description
RCU_TIMER12RST	TIMER12 clock reset
RCU_TIMER13RST	TIMER13 clock reset
RCU_WWDGTRST	WWDGT clock reset
RCU_SPI1RST	SPI1 clock reset
RCU_SPI2RST	SPI2 clock reset
RCU_USART1RST	USART1 clock reset
RCU_USART2RST	USART2 clock reset
RCU_UART3RST	UART3 clock reset
RCU_UART4RST	UART4 clock reset
RCU_I2C0RST	I2C0 clock reset
RCU_I2C1RST	I2C1 clock reset
RCU_USBDRST	USBD clock reset
RCU_I2C2RST	I2C2 clock reset
RCU_BKPIRST	BKPI clock reset
RCU_PMURST	PMU clock reset
RCU_DACRST	DAC clock reset
RCU_CTCRST	CTC clock reset
RCU_AFRST	alternate function clock reset
RCU_GPIOARST	GPIOA clock reset
RCU_GPIOBRST	GPIOB clock reset
RCU_GPIOCRST	GPIOC clock reset
RCU_GPIODRST	GPIOD clock reset
RCU_GPIOERST	GPIOE clock reset
RCU_ADC0RST	ADC0 clock reset
RCU_ADC1RST	ADC1 clock reset
RCU_TIMER0RST	TIMER0 clock reset
RCU_SPI0RST	SPI0 clock reset
RCU_TIMER7RST	TIMER7 clock reset
RCU_USART0RST	USART0 clock reset
RCU_TIMER8RST	TIMER8 clock reset
RCU_TIMER9RST	TIMER9 clock reset
RCU_TIMER10RST	TIMER10 clock reset

Enum rcu_flag_enum

Table 3-327. rcu_flag_enum

enum name	Function description
RCU_FLAG_IRC8MSTB	IRC8M stabilization flags
RCU_FLAG_HXTALSTB	HXTAL stabilization flags
RCU_FLAG_PLLSTB	PLL stabilization flags

enum name	Function description
RCU_FLAG_PLL1STB	PLL1 stabilization flags
RCU_FLAG_PLL2STB	PLL2 stabilization flags
RCU_FLAG_LXTALSTB	LXTAL stabilization flags
RCU_FLAG_IRC40KSTB	IRC40K stabilization flags
RCU_FLAG_IRC48MSTB	IRC48M stabilization flags
RCU_FLAG_EPRST	external PIN reset flags
RCU_FLAG_PORRST	power reset flags
RCU_FLAG_SWRST	software reset flags
RCU_FLAG_FWDGTRST	FWDGT reset flags
RCU_FLAG_WWDGTRST	WWDGT reset flags
RCU_FLAG_LPRST	low-power reset flags

Enum rcu_int_flag_enum

Table 3-328. rcu_int_flag_enum

enum name	Function description
RCU_INT_FLAG_IRC40KSTB	IRC40K stabilization interrupt flag
RCU_INT_FLAG_LXTALSTB	LXTAL stabilization interrupt flag
RCU_INT_FLAG_IRC8MSTB	IRC8M stabilization interrupt flag
RCU_INT_FLAG_HXTALSTB	HXTAL stabilization interrupt flag
RCU_INT_FLAG_PLLSTB	PLL stabilization interrupt flag
RCU_INT_FLAG_PLL1STB	PLL1 stabilization interrupt flag
RCU_INT_FLAG_PLL2STB	PLL2 stabilization interrupt flag
RCU_INT_FLAG_CKM	HXTAL clock stuck interrupt flag
RCU_INT_FLAG_IRC48MSTB	IRC48M stabilization interrupt flag

Enum rcu_int_flag_clear_enum

Table 3-329. rcu_int_flag_clear_enum

enum name	Function description
RCU_INT_FLAG_IRC40KSTB_CLR	IRC40K stabilization interrupt flags clear
RCU_INT_FLAG_LXTALSTB_CLR	LXTAL stabilization interrupt flags clear
RCU_INT_FLAG_IRC8MSTB_CLR	IRC8M stabilization interrupt flags clear
RCU_INT_FLAG_HXTALSTB_CLR	HXTAL stabilization interrupt flags clear
RCU_INT_FLAG_PLLSTB_CLR	PLL stabilization interrupt flags clear
RCU_INT_FLAG_PLL1STB_CLR	PLL1 stabilization interrupt flags clear
RCU_INT_FLAG_PLL2STB_CLR	PLL2 stabilization interrupt flags clear
RCU_INT_FLAG_CKM_CLR	CKM interrupt flags clear
RCU_INT_FLAG_IRC48MSTB_CLR	internal 48 MHz RC oscillator stabilization interrupt clear

Enum rcu_int_enum

Table 3-330. rcu_int_enum

enum name	Function description
RCU_INT_IRC40KSTB	IRC40K stabilization interrupt
RCU_INT_LXTALSTB	LXTAL stabilization interrupt
RCU_INT_IRC8MSTB	IRC8M stabilization interrupt
RCU_INT_HXTALSTB	HXTAL stabilization interrupt
RCU_INT_PLLSTB	PLL stabilization interrupt
RCU_INT_PLL1STB	PLL1 stabilization interrupt
RCU_INT_PLL2STB	PLL2 stabilization interrupt
RCU_INT_IRC48MSTB	internal 48 MHz RC oscillator stabilization interrupt

Enum rcu_osci_type_enum

Table 3-331. rcu_osci_type_enum

enum name	Function description
RCU_HXTAL	HXTAL
RCU_LXTAL	LXTAL
RCU_IRC8M	IRC8M
RCU_IRC48M	IRC48M

enum name	Function description
RCU_IRC40K	IRC40K
RCU_PLL_CK	PLL
RCU_PLL1_CK	PLL1
RCU_PLL2_CK	PLL2

Enum rcu_clock_freq_enum

Table 3-332. rcu_clock_freq_enum

enum name	Function description
CK_SYS	system clock
CK_AHB	AHB clock
CK_APB1	APB1 clock
CK_APB2	APB2 clock

rcu_deinit

The description of rcu_deinit is shown as below:

Table 3-333. Function rcu_deinit

Function name	rcu_deinit
Function prototype	void rcu_deinit(void);
Function descriptions	deinitialize the RCU, reset the value of all RCU registers into initial values
Precondition	-
The called functions	rcu_osci_stab_wait
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset RCU */

rcu_deinit();
```

rcu_periph_clock_enable

The description of rcu_periph_clock_enable is shown as below:

Table 3-334. Function rcu_periph_clock_enable

Function name	rcu_periph_clock_enable
Function prototype	void rcu_periph_clock_enable(rcu_periph_enum periph);
Function descriptions	enable the peripherals clock

Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to *Table 3-324. rcu_periph_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the USART0 clock */
rcu_periph_clock_enable(RCU_USART0);
```

rcu_periph_clock_disable

The description of rcu_periph_clock_disable is shown as below:

Table 3-335. Function rcu_periph_clock_disable

Function name	rcu_periph_clock_disable
Function prototype	void rcu_periph_clock_disable(rcu_periph_enum periph);
Function descriptions	disable the peripherals clock
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to *Table 3-324. rcu_periph_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the USART0 clock */
rcu_periph_clock_disable(RCU_USART0);
```

rcu_periph_clock_sleep_enable

The description of rcu_periph_clock_sleep_enable is shown as below:

Table 3-336. Function rcu_periph_clock_sleep_enable

Function name	rcu_periph_clock_sleep_enable
Function prototype	void rcu_periph_clock_sleep_enable(rcu_periph_sleep_enum periph);
Function descriptions	enable the peripherals clock when in sleep mode
Precondition	-

The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to *Table 3-325. rcu_periph_sleep_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the FMC clock when in sleep mode */
rcu_periph_clock_sleep_enable(RCU_FMC_SLP);
```

rcu_periph_clock_sleep_disable

The description of rcu_periph_clock_sleep_disable is shown as below:

Table 3-337. Function rcu_periph_clock_sleep_disable

Function name	rcu_periph_clock_sleep_disable
Function prototype	void rcu_periph_clock_sleep_disable(rcu_periph_sleep_enum periph);
Function descriptions	disable the peripherals clock when in sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to *Table 3-325. rcu_periph_sleep_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the FMC clock when in sleep mode */
rcu_periph_clock_sleep_disable(RCU_FMC_SLP);
```

rcu_periph_reset_enable

The description of rcu_periph_reset_enable is shown as below:

Table 3-338. Function rcu_periph_reset_enable

Function name	rcu_periph_reset_enable
Function prototype	void rcu_periph_reset_enable(rcu_periph_reset_enum periph_reset);
Function descriptions	enable the peripherals reset
Precondition	-
The called functions	-

Input parameter{in}	
periph_reset	RCU peripherals reset, refer to *Table 3-326. rcu_periph_reset_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 reset */
rcu_periph_reset_enable(RCU_SPI0RST);
```

rcu_periph_reset_disable

The description of rcu_periph_reset_disable is shown as below:

Table 3-339. Function rcu_periph_reset_disable

Function name	rcu_periph_reset_disable
Function prototype	void rcu_periph_reset_disable(rcu_periph_reset_enum periph_reset);
Function descriptions	disable the peripheral reset
Precondition	-
The called functions	-
Input parameter{in}	
periph_reset	RCU peripherals reset, refer to *Table 3-326. rcu_periph_reset_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 reset */
rcu_periph_reset_disable(RCU_SPI0RST);
```

rcu_bkp_reset_enable

The description of rcu_bkp_reset_enable is shown as below:

Table 3-340. Function rcu_bkp_reset_enable

Function name	rcu_bkp_reset_enable
Function prototype	void rcu_bkp_reset_enable(void);
Function descriptions	enable the BKP domain reset
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* reset the BKP domain */

rcu_bkp_reset_enable();

rcu_bkp_reset_disable

The description of rcu_bkp_reset_disable is shown as below:

Table 3-341. Function rcu_bkp_reset_disable

Function name	rcu_bkp_reset_disable
Function prototype	void rcu_bkp_reset_disable(void);
Function descriptions	disable the BKP domain reset
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the BKP domain reset */

rcu_bkp_reset_disable();

rcu_system_clock_source_config

The description of rcu_system_clock_source_config is shown as below:

Table 3-342. Function rcu_system_clock_source_config

Function name	rcu_system_clock_source_config
Function prototype	void rcu_system_clock_source_config(uint32_t ck_sys);
Function descriptions	configure the system clock source
Precondition	-
The called functions	-
Input parameter{in}	
ck_sys	system clock source select

RCU_CKSYSSRC_IRC8M	select CK_IRC8M as the CK_SYS source
RCU_CKSYSSRC_HXTAL	select CK_HXTAL as the CK_SYS source
RCU_CKSYSSRC_PLL	select CK_PLL as the CK_SYS source
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_HXTAL as the CK_SYS source */
rcu_system_clock_source_config(RCU_CKSYSSRC_HXTAL);
```

rcu_system_clock_source_get

The description of rcu_system_clock_source_get is shown as below:

Table 3-343. Function rcu_system_clock_source_get

Function name	rcu_system_clock_source_get
Function prototype	uint32_t rcu_system_clock_source_get(void);
Function descriptions	get the system clock source
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	RCU_SCSS_IRC8M/RCU_SCSS_HXTAL/RCU_SCSS_PLL

Example:

```
uint32_t temp_cksys_status;
/* get the CK_SYS source */
temp_cksys_status = rcu_system_clock_source_get();
```

rcu_ahb_clock_config

The description of rcu_ahb_clock_config is shown as below:

Table 3-344. Function rcu_ahb_clock_config

Function name	rcu_ahb_clock_config
Function prototype	void rcu_ahb_clock_config(uint32_t ck_ahb);

Function descriptions	configure the AHB clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_ahb	AHB clock prescaler selection
RCU_AHB_CKSYS_DIVx	select CK_SYS / x, (x=1, 2, 4, 8, 16, 64, 128, 256, 512)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_SYS/128 */
rcu_ahb_clock_config(RCU_AHB_CKSYS_DIV128);
```

rcu_apb1_clock_config

The description of rcu_apb1_clock_config is shown as below:

Table 3-345. Function rcu_apb1_clock_config

Function name	rcu_apb1_clock_config
Function prototype	void rcu_apb1_clock_config(uint32_t ck_apb1);
Function descriptions	configure the APB1 clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_apb1	APB1 clock prescaler selection
RCU_APB1_CKAHB_DIV1	select CK_AHB as CK_APB1
RCU_APB1_CKAHB_DIV2	select CK_AHB/2 as CK_APB1
RCU_APB1_CKAHB_DIV4	select CK_AHB/4 as CK_APB1
RCU_APB1_CKAHB_DIV8	select CK_AHB/8 as CK_APB1
RCU_APB1_CKAHB_DIV16	select CK_AHB/16 as CK_APB1
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure CK_AHB/16 as CK_APB1 */

rcu_apb1_clock_config(RCU_APB1_CKAHB_DIV16);

rcu_apb2_clock_config

The description of rcu_apb2_clock_config is shown as below:

Table 3-346. Function rcu_apb2_clock_config

Function name	rcu_apb2_clock_config
Function prototype	void rcu_apb2_clock_config(uint32_t ck_apb2);
Function descriptions	configure the APB2 clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_apb2	APB2 clock prescaler selection
RCU_APB2_CKAHB_DIV1	select CK_AHB as CK_APB2
RCU_APB2_CKAHB_DIV2	select CK_AHB/2 as CK_APB2
RCU_APB2_CKAHB_DIV4	select CK_AHB/4 as CK_APB2
RCU_APB2_CKAHB_DIV8	select CK_AHB/8 as CK_APB2
RCU_APB2_CKAHB_DIV16	select CK_AHB/16 as CK_APB2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure CK_AHB/8 as CK_APB2 */

rcu_apb2_clock_config(RCU_APB2_CKAHB_DIV8);

rcu_ckout0_config

The description of rcu_ckout0_config is shown as below:

Table 3-347. Function rcu_ckout0_config

Function name	rcu_ckout0_config
Function prototype	void rcu_ckout0_config(uint32_t ckout0_src);
Function descriptions	configure the CK_OUT0 clock source
Precondition	-

The called functions	-
Input parameter{in}	
ckout0_src	CK_OUT0 clock source selection
RCU_CKOUT0SRC_NONE	no clock selected
RCU_CKOUT0SRC_CKSYS	select system clock CK_SYS
RCU_CKOUT0SRC_IRC8M	select high speed 8M internal oscillator clock
RCU_CKOUT0SRC_HXTAL	select HXTAL
RCU_CKOUT0SRC_CKPLL_DIV2	select (CK_PLL / 2) clock
RCU_CKOUT0SRC_CKPLL1	select CK_PLL1 clock
RCU_CKOUT0SRC_CKPLL2_DIV2	select (CK_PLL2 / 2) clock
RCU_CKOUT0SRC_CKPLL2	select CK_PLL2 clock
RCU_CKOUT0SRC_IRC48M	select IRC48M clock
RCU_CKOUT0SRC_IRC8M_DIV8	select (IRC48M / 8) clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the HXTAL as CK_OUT0 clock source */
rcu_ckout0_config(RCU_CKOUT0SRC_HXTAL);
```

rcu_pll_config

The description of rcu_pll_config is shown as below:

Table 3-348. Function rcu_pll_config

Function name	rcu_pll_config
Function prototype	void rcu_pll_config(uint32_t pll_src, uint32_t pll_mul);
Function descriptions	configure the main PLL clock
Precondition	-
The called functions	-
Input parameter{in}	

pll_src	PLL clock source selection
RCU_PLLSRC_IRC8M_DIV2	IRC8M/2 clock is selected as source clock of PLL
RCU_PLLSRC_HXTAL_IRC48M	HXTAL or IRC48M is selected as source clock of PLL
Input parameter{in}	
pll_mul	PLL clock multiplication factor
RCU_PLL_MULx	PLL clock * x (x = 2..14, 6.5, 16..31)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PLL */
rcu_pll_config(RCU_PLLSRC_HXTAL, RCU_PLL_MUL10);
```

rcu_pllpresel_config

The description of rcu_pllpresel_config is shown as below:

Table 3-349. Function rcu_pllpresel_config

Function name	rcu_pllpresel_config
Function prototype	void rcu_pllpresel_config(uint32_t pll_presel);
Function descriptions	configure the PLL clock source preselection
Precondition	-
The called functions	-
Input parameter{in}	
pll_presel	PLL clock source preselection
RCU_PLLPRESRC_HXTAL	HXTAL selected as PREDV0 source clock
RCU_PLLPRESRC_IRC48M	CK_PLL selected as PREDV0 input source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PLL clock source preselection */
rcu_pllpresel_config (RCU_PLLPRESRC_HXTAL);
```

rcu_predv0_config

The description of rcu_predv0_config is shown as below:

Table 3-350. Function rcu_predv0_config

Function name	rcu_predv0_config
Function prototype	void rcu_predv0_config(uint32_t predv0_source, uint32_t predv0_div);
Function descriptions	configure the PREDV0 division factor
Precondition	-
The called functions	-
Input parameter{in}	
predv0_source	PREDV0 input clock source selection
RCU_PREDV0SRC_HXTAL_IRC48M	select HXTAL or IRC48M as PREDV0 input source clock
RCU_PREDV0SRC_CKPLL1	select CK_PLL1 as PREDV0 input source clock
Input parameter{in}	
predv0_div	PREDV0 division factor
RCU_PREDV0_DIVx	PREDV0 input source clock is divided x (x=1..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PREDV0 division factor */
rcu_predv0_config(RCU_PREDV0SRC_HXTAL_IRC48M, RCU_PREDV0_DIV4);
```

rcu_predv1_config

The description of rcu_predv1_config is shown as below:

Table 3-351. Function rcu_predv1_config

Function name	rcu_predv1_config
Function prototype	void rcu_predv1_config(uint32_t predv1_div);
Function descriptions	configure the PREDV1 division factor
Precondition	-
The called functions	-
Input parameter{in}	
predv1_div	PREDV1 division factor
RCU_PREDV1_DIVx	PREDV1 input source clock is divided x (x=1..16)
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* configure the PREDV1 division factor */

rcu_predv1_config(RCU_PREDV1_DIV8);

rcu_pll1_config

The description of rcu_pll1_config is shown as below:

Table 3-352. Function rcu_pll1_config

Function name	rcu_pll1_config
Function prototype	void rcu_pll1_config(uint32_t pll_mul);
Function descriptions	configure the PLL1 clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_mul	PLL clock multiplication factor
RCU_PLL1_MULx	PLL1 clock * x，(x = 8..16, 20)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the PLL1 clock */

rcu_pll1_config(RCU_PLL1_MUL8);

rcu_pll2_config

The description of rcu_pll2_config is shown as below:

Table 3-353. Function rcu_pll2_config

Function name	rcu_pll2_config
Function prototype	void rcu_pll2_config(uint32_t pll_mul)
Function descriptions	configure the PLL2 clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_mul	PLL clock multiplication factor
RCU_PLL2_MULx	PLL2 clock * x, (x = 8..16, 20)
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* configure the PLL2 clock */

rcu_pll2_config(RCU_PLL2_MUL8);

rcu_adc_clock_config

The description of rcu_adc_clock_config is shown as below:

Table 3-354. Function rcu_adc_clock_config

Function name	rcu_adc_clock_config
Function prototype	void rcu_adc_clock_config(uint32_t adc_psc);
Function descriptions	configure the ADC prescaler factor
Precondition	-
The called functions	-
Input parameter{in}	
adc_psc	ADC prescaler factor
RCU_CKADC_CKAPB2_DIV2	CK_ADC = CK_APB2 / 2
RCU_CKADC_CKAPB2_DIV4	CK_ADC = CK_APB2 / 4
RCU_CKADC_CKAPB2_DIV6	CK_ADC = CK_APB2 / 6
RCU_CKADC_CKAPB2_DIV8	CK_ADC = CK_APB2 / 8
RCU_CKADC_CKAPB2_DIV12	CK_ADC = CK_APB2 / 12
RCU_CKADC_CKAPB2_DIV16	CK_ADC = CK_APB2 / 16
RCU_CKADC_CKAHB_DIV3	CK_ADC = CK_AHB/ 3
RCU_CKADC_CKAHB_DIV5	CK_ADC = CK_AHB/ 5
RCU_CKADC_CKAHB_DIV7	CK_ADC = CK_AHB/ 7
RCU_CKADC_CKAHB_DIV9	CK_ADC = CK_AHB/ 9
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the ADC prescaler factor */

rcu_adc_clock_config(RCU_CKADC_CKAPB2_DIV8);

rcu_usb_clock_config

The description of rcu_usb_clock_config is shown as below:

Table 3-355. Function rcu_usb_clock_config

Function name	rcu_usb_clock_config
Function prototype	void rcu_usb_clock_config(uint32_t usb_psc);
Function descriptions	configure the USB prescaler factor
Precondition	-
The called functions	-
Input parameter{in}	
usb_psc	USB prescaler factor
RCU_CKUSB_CKPLL_DIV1_5	CK_USBFS = CK_PLL / 1.5
RCU_CKUSB_CKPLL_DIV1	CK_USBFS = CK_PLL / 1
RCU_CKUSB_CKPLL_DIV2_5	CK_USBFS = CK_PLL / 2.5
RCU_CKUSB_CKPLL_DIV2	CK_USBFS = CK_PLL / 2
RCU_CKUSB_CKPLL_DIV3	CK_USBFS = CK_PLL / 3
RCU_CKUSB_CKPLL_DIV3_5	CK_USBFS = CK_PLL / 3.5
RCU_CKUSB_CKPLL_DIV4	CK_USBFS = CK_PLL / 4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the USB prescaler factor */

rcu_usb_clock_config(RCU_CKUSB_CKPLL_DIV2_5);

rcu_rtc_clock_config

The description of rcu_rtc_clock_config is shown as below:

Table 3-356. Function rcu_rtc_clock_config

Function name	rcu_rtc_clock_config
Function prototype	void rcu_rtc_clock_config(uint32_t rtc_clock_source);
Function descriptions	configure the RTC clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
rtc_clock_source	RTC clock source selection
RCU_RTCSRC_NONE	no clock selected
RCU_RTCSRC_LXTAL	select CK_LXTAL as RTC source clock
RCU_RTCSRC_IRC40K	select CK_IRC40K as RTC source clock
RCU_RTCSRC_HXTAL_DIV_128	select CK_HXTAL/128 as RTC source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the RTC clock source selection */
rcu_rtc_clock_config(RCU_RTCSRC_IRC40K);
```

rcu_i2s1_clock_config

The description of rcu_i2s1_clock_config is shown as below:

Table 3-357. Function rcu_i2s1_clock_config

Function name	rcu_i2s1_clock_config
Function prototype	void rcu_i2s1_clock_config(uint32_t i2s_clock_source);
Function descriptions	configure the I2S1 clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
i2s_clock_source	I2S clock source selection
RCU_I2S1SRC_CKSYS	select system clock as I2S1 source clock
RCU_I2S1SRC_CKPLL2_MUL2	select CK_PLL2 * 2 as I2S1 source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the I2S1 clock source selection */

rcu_i2s1_clock_config(RCU_I2S1SRC_CKPLL2_MUL2);

rcu_i2s2_clock_config

The description of rcu_i2s2_clock_config is shown as below:

Table 3-358. Function rcu_i2s2_clock_config

Function name	rcu_i2s2_clock_config
Function prototype	void rcu_i2s2_clock_config(uint32_t i2s_clock_source);
Function descriptions	configure the I2S2 clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
i2s_clock_source	I2S clock source selection
RCU_I2S2SRC_CKSYS	select system clock as I2S2 source clock
RCU_I2S2SRC_CKPLL2_MUL2	select CK_PLL2 * 2 as I2S2 source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the I2S2 clock source selection */

rcu_i2s2_clock_config(RCU_I2S2SRC_CKPLL2_MUL2);

rcu_ck48m_clock_config

The description of rcu_ck48m_clock_config is shown as below:

Table 3-359. Function rcu_ck48m_clock_config

Function name	rcu_ck48m_clock_config
Function prototype	void rcu_ck48m_clock_config(uint32_t ck48m_clock_source);
Function descriptions	configure the CK48M clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
ck48m_clock_source	CK48M clock source selection
RCU_CK48MSRC_CKPLL	CK_PLL selected as CK48M source clock

RCU_CK48MSRC_IRC48M	CK_IRC48M selected as CK48M source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK48M clock source selection */
rcu_ck48m_clock_config(RCU_CK48MSRC_CKPLL);
```

rcu_flag_get

The description of rcu_flag_get is shown as below:

Table 3-360. Function rcu_flag_get

Function name	rcu_flag_get
Function prototype	FlagStatus rcu_flag_get(rcu_flag_enum flag);
Function descriptions	get the clock stabilization and periphral reset flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	the clock stabilization and periphral reset flags, refer to *Table 3-327. rcu_flag_enum*
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the clock stabilization flag */
if(RESET != rcu_flag_get(RCU_FLAG_LXTALSTB)){
}
```

rcu_all_reset_flag_clear

The description of rcu_all_reset_flag_clear is shown as below:

Table 3-361. Function rcu_all_reset_flag_clear

Function name	rcu_all_reset_flag_clear
Function prototype	void rcu_all_reset_flag_clear(void);
Function descriptions	clear all the reset flag
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear all the reset flag */
rcu_all_reset_flag_clear();
```

rcu_interrupt_flag_get

The description of rcu_interrupt_flag_get is shown as below:

Table 3-362. Function rcu_interrupt_flag_get

Function name	rcu_interrupt_flag_get
Function prototype	FlagStatus rcu_interrupt_flag_get(rcu_int_flag_enum int_flag);
Function descriptions	get the clock stabilization interrupt and ckm flags
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt and ckm flags, refer to *Table 3-328. rcu_int_flag_enum*
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the clock stabilization interrupt flag */
if(SET == rcu_interrupt_flag_get(RCU_INT_FLAG_HXTALSTB)){
}
```

rcu_interrupt_flag_clear

The description of rcu_interrupt_flag_clear is shown as below:

Table 3-363. Function rcu_interrupt_flag_clear

Function name	rcu_interrupt_flag_clear
Function prototype	void rcu_interrupt_flag_clear(rcu_int_flag_clear_enum int_flag_clear)
Function descriptions	clear the interrupt flags
Precondition	-

The called functions	-
Input parameter{in}	
int_flag_clear	clock stabilization and stuck interrupt flags clear, refer to *Table 3-329. rcu_int_flag_clear_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the interrupt HXTAL stabilization interrupt flag */
rcu_interrupt_flag_clear(RCU_INT_FLAG_HXTALSTB_CLR);
```

rcu_interrupt_enable

The description of rcu_interrupt_enable is shown as below:

Table 3-364. Function rcu_interrupt_enable

Function name	rcu_interrupt_enable
Function prototype	void rcu_interrupt_enable(rcu_int_enum stab_int);
Function descriptions	enable the stabilization interrupt
Precondition	-
The called functions	-
Input parameter{in}	
stab_int	clock stabilization interrupt, refer to *Table 3-330. rcu_int_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the HXTAL stabilization interrupt */
rcu_interrupt_enable(RCU_INT_HXTALSTB);
```

rcu_interrupt_disable

The description of rcu_interrupt_disable is shown as below:

Table 3-365. Function rcu_interrupt_disable

Function name	rcu_interrupt_disable
Function prototype	void rcu_interrupt_disable(rcu_int_enum stab_int);
Function descriptions	disable the stabilization interrupt
Precondition	-

The called functions	-
Input parameter{in}	
stab_int	clock stabilization interrupt, refer to *Table 3-330. rcu_int_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the HXTAL stabilization interrupt */
rcu_interrupt_disable(RCU_INT_HXTALSTB);
```

rcu_lxtal_drive_capability_config

The description of rcu_lxtal_drive_capability_config is shown as below:

Table 3-366. Function rcu_lxtal_drive_capability_config

Function name	rcu_lxtal_drive_capability_config
Function prototype	void rcu_lxtal_drive_capability_config(uint32_t lxtal_dricap);
Function descriptions	configure the LXTAL drive capability
Precondition	-
The called functions	-
Input parameter{in}	
lxtal_dricap	drive capability of LXTAL
RCU_LXTAL_LOWDRI	lower driving capability
RCU_LXTAL_MED_LOWDRI	medium low driving capability
RCU_LXTAL_MED_HIGHDRI	medium high driving capability
RCU_LXTAL_HIGHDRI	higher driving capability
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the LXTAL drive capability */
rcu_lxtal_drive_capability_config (RCU_LXTAL_LOWDRI);
```

rcu_osci_stab_wait

The description of rcu_osci_stab_wait is shown as below:

Table 3-367. Function rcu_osci_stab_wait

Function name	rcu_osci_stab_wait
Function prototype	ErrStatus rcu_osci_stab_wait(rcu_osci_type_enum osci);
Function descriptions	wait for oscillator stabilization flags is SET or oscillator startup is timeout
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to *Table 3-331. rcu_osci_type_enum*
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* wait for oscillator stabilization flag */
if(SUCCESS == rcu_osci_stab_wait(RCU_HXTAL)){
}
```

rcu_osci_on

The description of rcu_osci_on is shown as below:

Table 3-368. Function rcu_osci_on

Function name	rcu_osci_on
Function prototype	void rcu_osci_on(rcu_osci_type_enum osci);
Function descriptions	turn on the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to *Table 3-331. rcu_osci_type_enum*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn on the high speed crystal oscillator */
rcu_osci_on(RCU_HXTAL);
```

rcu_osci_off

The description of rcu_osci_off is shown as below:

Table 3-369. Function rcu_osci_off

Function name	rcu_osci_off
Function prototype	void rcu_osci_off(rcu_osci_type_enum osci);
Function descriptions	turn off the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to ***Table 3-331. rcu_osci_type_enum***
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn off the high speed crystal oscillator */
rcu_osci_off(RCU_HXTAL);
```

rcu_osci_bypass_mode_enable

The description of rcu_osci_bypass_mode_enable is shown as below:

Table 3-370. Function rcu_osci_bypass_mode_enable

Function name	rcu_osci_bypass_mode_enable
Function prototype	void rcu_osci_bypass_mode_enable(rcu_osci_type_enum osci);
Function descriptions	enable the oscillator bypass mode
Precondition	HXTALEN or LXTALEN must be reset before it
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to ***Table 3-331. rcu_osci_type_enum***
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the high speed crystal oscillator bypass mode */
rcu_osci_bypass_mode_enable(RCU_HXTAL);
```

rcu_osci_bypass_mode_disable

The description of rcu_osci_bypass_mode_disable is shown as below:

Table 3-371. Function rcu_osci_bypass_mode_disable

Function name	rcu_osci_bypass_mode_disable
Function prototype	void rcu_osci_bypass_mode_disable(rcu_osci_type_enum osci);
Function descriptions	disable the oscillator bypass mode
Precondition	HXTALEN or LXTALEN must be reset before it
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to *Table 3-331. rcu_osci_type_enum*
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_LXTAL	low speed crystal oscillator(LXTAL)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the high speed crystal oscillator bypass mode */
rcu_osci_bypass_mode_disable(RCU_HXTAL);
```

rcu_hxtal_clock_monitor_enable

The description of rcu_hxtal_clock_monitor_enable is shown as below:

Table 3-372. Function rcu_hxtal_clock_monitor_enable

Function name	rcu_hxtal_clock_monitor_enable
Function prototype	void rcu_hxtal_clock_monitor_enable(void);
Function descriptions	enable the HXTAL clock monitor
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the HXTAL clock monitor */
rcu_hxtal_clock_monitor_enable();
```

rcu_hxtal_clock_monitor_disable

The description of rcu_hxtal_clock_monitor_disable is shown as below:

Table 3-373. Function rcu_hxtal_clock_monitor_disable

Function name	rcu_hxtal_clock_monitor_disable
Function prototype	void rcu_hxtal_clock_monitor_disable(void);
Function descriptions	disable the HXTAL clock monitor
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the HXTAL clock monitor */
rcu_hxtal_clock_monitor_disable();
```

rcu_irc8m_adjust_value_set

The description of rcu_irc8m_adjust_value_set is shown as below:

Table 3-374. Function rcu_irc8m_adjust_value_set

Function name	rcu_irc8m_adjust_value_set
Function prototype	void rcu_irc8m_adjust_value_set(uint32_t irc8m_adjval);
Function descriptions	set the IRC8M adjust value
Precondition	-
The called functions	-
Input parameter{in}	
irc8m_adjval	IRC8M adjust value, must be between 0 and 0x1F
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the IRC8M adjust value */
rcu_irc8m_adjust_value_set(0x10);
```

rcu_deepsleep_voltage_set

The description of rcu_deepsleep_voltage_set is shown as below:

Table 3-375. Function rcu_deepsleep_voltage_set

Function name	rcu_deepsleep_voltage_set
Function prototype	void rcu_deepsleep_voltage_set(uint32_t dsvol);
Function descriptions	set the deep-sleep mode voltage value
Precondition	-
The called functions	-
Input parameter{in}	
dsvol	deep sleep mode voltage
RCU_DEEPSLEEP_V_1_0	the core voltage is 1.0V in deep-sleep mode
RCU_DEEPSLEEP_V_0_9	the core voltage is 0.9V in deep-sleep mode
RCU_DEEPSLEEP_V_0_8	the core voltage is 0.8V in deep-sleep mode
RCU_DEEPSLEEP_V_1_2	the core voltage is 1.2V in deep-sleep mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the deep-sleep mode voltage */
rcu_deepsleep_voltage_set(RCU_DEEPSLEEP_V_1_0);
```

rcu_clock_freq_get

The description of rcu_clock_freq_get is shown as below:

Table 3-376. Function rcu_clock_freq_get

Function name	rcu_clock_freq_get
Function prototype	uint32_t rcu_clock_freq_get(rcu_clock_freq_enum clock);
Function descriptions	get the system clock, bus clock frequency
Precondition	-
The called functions	-
Input parameter{in}	
clock	the clock frequency which to get
CK_SYS	system clock frequency
CK_AHB	AHB clock frequency
CK_APB1	APB1 clock frequency
CK_APB2	APB2 clock frequency
Output parameter{out}	
-	-

Return value	
ck_freq	clock frequency of system, AHB, APB1, APB2

Example:

```
uint32_t temp_freq;

/* get the system clock frequency */

temp_freq = rcu_clock_freq_get(CK_SYS);
```

3.18. RTC

The Real-time Clock (RTC) is usually used as a clock-calendar. The ones in the Backup Domain consist of a 32-bit up-counter, an alarm, a prescaler, a divider and the RTC clock configuration register. The RTC registers are listed in chapter *3.18.1*, the RTC firmware functions are introduced in chapter *3.18.2*.

3.18.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

Table 3-377. RTC Registers

Registers	Descriptions
RTC_INTEN	Interrupt enable register
RTC_CTL	Control register
RTC_PSCH	Prescaler high register
RTC_PSCL	Prescaler low register
RTC_DIVH	Divider high register
RTC_DIVL	Divider low register
RTC_CNTH	counter high register
RTC_CNTL	counter low register
RTC_ALRMH	Alarm high register
RTC_ALRML	Alarm low register

3.18.2. Descriptions of Peripheral functions

RTC firmware functions are listed in the table shown as below:

Table 3-378. RTC firmware function

Function name	Function description
rtc_configuration_mode_enter	enter RTC configuration mode
rtc_configuration_mode_exit	exit RTC configuration mode
rtc_counter_set	set RTC counter value
rtc_prescaler_set	set RTC prescaler value

Function name	Function description
rtc_lwoff_wait	wait RTC last write operation finished flag set
rtc_register_sync_wait	wait RTC registers synchronized flag set
rtc_alarm_config	set RTC alarm value
rtc_counter_get	get RTC counter value
rtc_divider_get	get RTC divider value
rtc_flag_get	get RTC flag status
rtc_flag_clear	clear RTC flag status
rtc_interrupt_flag_get	get RTC interrupt flag status
rtc_interrupt_flag_clear	clear RTC interrupt flag status
rtc_interrupt_enable	enable RTC interrupt
rtc_interrupt_disable	disable RTC interrupt

rtc_configuration_mode_enter

The description of rtc_configuration_mode_enter is shown as below:

Table 3-379. Function rtc_configuration_mode_enter

Function name	rtc_configuration_mode_enter
Function prototype	void rtc_configuration_mode_enter(void);
Function descriptions	enter RTC configuration mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enter RTC configuration mode */
rtc_configuration_mode_enter( );
```

rtc_configuration_mode_exit

The description of rtc_configuration_mode_exit is shown as below:

Table 3-380. Function rtc_configuration_mode_exit

Function name	rtc_configuration_mode_exit
Function prototype	void rtc_configuration_mode_exit(void);
Function descriptions	exit RTC configuration mode
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* exit RTC configuration mode */
rtc_configuration_mode_exit ( );
```

rtc_counter_set

The description of rtc_counter_set is shown as below:

Table 3-381. Function rtc_counter_set

Function name	rtc_counter_set
Function prototype	void rtc_counter_set(uint32_t cnt);
Function descriptions	set RTC counter value
Precondition	-
The called functions	-
Input parameter{in}	
cnt	RTC counter value(0-0xFFFF FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* set counter value to 0xFFFF */
rtc_counter_set (0xFFFF);
```

rtc_prescaler_set

The description of rtc_prescaler_set is shown as below:

Table 3-382. Function rtc_prescaler_set

Function name	rtc_interrupt_rtc_prescaler_set
Function prototype	void rtc_prescaler_set(uint32_t psc);
Function descriptions	set RTC prescaler value

Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).
The called functions	rtc_configuration_mode_enter / rtc_configuration_mode_exit
Input parameter{in}	
psc	RTC prescaler value(0-0x000F FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* set RTC prescaler value to 0x7FFFF */
rtc_prescaler_set (0x7FFFF);
```

rtc_lwoff_wait

The description of rtc_lwoff_wait is shown as below:

Table 3-383. Function rtc_lwoff_wait

Function name	rtc_lwoff_wait
Function prototype	void rtc_lwoff_wait(void);
Function descriptions	wait RTC last write operation finished flag set
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* enable the RTC second interrupt */
rtc_interrupt_enable(RTC_INT_SECOND);
```

rtc_register_sync_wait

The description of rtc_register_sync_wait is shown as below:

Table 3-384. Function rtc_register_sync_wait

Function name	rtc_register_sync_wait
Function prototype	void rtc_register_sync_wait(void);
Function descriptions	wait RTC registers synchronized flag set
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait for RTC registers synchronization */
rtc_register_sync_wait( );
```

rtc_alarm_config

The description of rtc_alarm_config is shown as below:

Table 3-385. Function rtc_alarm_config

Function name	rtc_alarm_config
Function prototype	void rtc_alarm_config(uint32_t alarm);
Function descriptions	set RTC alarm value
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set). -
The called functions	rtc_configuration_mode_enter / rtc_configuration_mode_exit
Input parameter{in}	
alarm	RTC alarm value(0-0xFFFF FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* set alarm value to 0xFFFF */
```

rtc_alarm_config (0xFFFF);

rtc_counter_get

The description of rtc_counter_get is shown as below:

Table 3-386. Function rtc_counter_get

Function name	rtc_counter_get
Function prototype	uint32_t rtc_counter_get(void);
Function descriptions	get RTC counter value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the value of RTC counter

Example:

```
/* get the counter value */
uint32_t rtc_counter_value;
rtc_counter_value = rtc_counter_get ( );
```

rtc_divider_get

The description of rtc_divider_get is shown as below:

Table 3-387. Function rtc_divider_get

Function name	rtc_divider_get
Function prototype	uint32_t rtc_divider_get(void);
Function descriptions	get RTC divider value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the value of RTC divider

Example:

```
/* get the current RTC divider value */
```

```
uint32_t rtc_divider_value;

rtc_divider_value = rtc_divider_get ( );
```

rtc_flag_get

The description of rtc_flag_getrtc_interrupt_enable is shown as below:

Table 3-388. Function rtc_flag_get

Function name	rtc_flag_get
Function prototype	FlagStatus rtc_flag_get(uint32_t flag);
Function descriptions	get RTC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC flag status to get
RTC_FLAG_SECOND	second interrupt flag
RTC_FLAG_ALARM	alarm interrupt flag
RTC_FLAG_OVERFLOW	overflow interrupt flag
RTC_FLAG_RSYN	registers synchronized flag
RTC_FLAG_LWOF	last write operation finished flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the RTC overflow interrupt status */

FlagStatus alarm_status;

alarm_status = rtc_flag_get (RTC_FLAG_ALARM);
```

rtc_flag_clear

The description of rtc_flag_clear is shown as below:

Table 3-389. Function rtc_flag_clear

Function name	rtc_flag_clear
Function prototype	void rtc_flag_clear(uint32_t flag);
Function descriptions	clear RTC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC flag status to clear

RTC_FLAG_SECOND	second interrupt flag
RTC_FLAG_ALARM	alarm interrupt flag
RTC_FLAG_OVERFLOW	overflow interrupt flag
RTC_FLAG_RSYN	registers synchronized flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the RTC alarm flag */
rtc_flag_clear (RTC_FLAG_ALARM);
```

rtc_ interrupt_flag_get

The description of rtc_ interrupt_flag_getrtc_interrupt_enable is shown as below:

Table 3-390. Function rtc_ interrupt_flag_get

Function name	rtc_ interrupt_flag_get
Function prototype	FlagStatus rtc_ interrupt_flag_get(uint32_t flag);
Function descriptions	get RTC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC interrupt flag status to get
RTC_INT_FLAG_SECOND	second interrupt flag
RTC_INT_FLAG_ALARM	alarm interrupt flag
RTC_INT_FLAG_OVERFLOW	overflow interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the RTC alarm interrupt status */
FlagStatus alarm_status;
alarm_status = rtc_interrupt_flag_get (RTC_INT_FLAG_ALARM);
```

rtc_interrupt_flag_clear

The description of rtc_interrupt_flag_clear is shown as below:

Table 3-391. Function rtc_interrupt_flag_clear

Function name	rtc_interrupt_flag_clear
Function prototype	void rtc_interrupt_flag_clear(uint32_t flag);
Function descriptions	clear RTC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC interrupt flag status to clear
RTC_INT_FLAG_SECOND	second interrupt flag
RTC_INT_FLAG_ALARM	alarm interrupt flag
RTC_INT_FLAG_OVERFLOW	overflow interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the RTC alarm interrupt flag */
rtc_interrupt_flag_clear (RTC_INT_FLAG_ALARM);
```

rtc_interrupt_enable

The description of rtc_interrupt_enable is shown as below:

Table 3-392. Function rtc_interrupt_enable

Function name	rtc_interrupt_enable
Function prototype	void rtc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable RTC interrupt
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).
The called functions	-
Input parameter{in}	
interrupt	specify which RTC interrupt to enable
RTC_INT_SECOND	second interrupt
RTC_INT_ALARM	alarm interrupt
RTC_INT_OVERFLOW	overflow interrupt
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* enable the RTC second interrupt */
rtc_interrupt_enable(RTC_INT_SECOND);
```

rtc_interrupt_disable

The description of rtc_interrupt_disable is shown as below:

Table 3-393. Function rtc_interrupt_disable

Function name	rtc_interrupt_disable
Function prototype	void rtc_interrupt_disable(uint32_t interrupt);
Function descriptions	disable RTC interrupt
Precondition	before using this function, you must call rtc_lwoff_wait ( ) function (wait until LWOFF flag is set).
The called functions	-
Input parameter{in}	
interrupt	specify which RTC interrupt to disable
RTC_INT_SECOND	second interrupt
RTC_INT_ALARM	alarm interrupt
RTC_INT_OVERFLOW	overflow interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait( );
/* disable the RTC second interrupt */
rtc_interrupt_disable(RTC_INT_SECOND);
```

3.19. SPI

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S

audio protocol. The SPI/I2S registers are listed in chapter *3.19.1*, the SPI/I2S firmware functions are introduced in chapter *3.19.2*.

3.19.1. Descriptions of Peripheral registers

SPI/I2S registers are listed in the table shown as below:

Table 3-394. SPI/I2S Registers

Registers	Descriptions
SPI_CTL0	SPI control register 0
SPI_CTL1	SPI control register 1
SPI_STAT	SPI status register
SPI_DATA	SPI data register
SPI_CRCPOLY	SPI CRC polynomial register
SPI_RCRC	SPI receive CRC register
SPI_TCRC	SPI transmit CRC register
SPI_I2SCTL	SPI/I2S control register
SPI_I2SPSC	SPI/I2S clock prescaler register
SPI_QCTL	Quad-SPI mode control register

3.19.2. Descriptions of Peripheral functions

SPI/I2S firmware functions are listed in the table shown as below:

Table 3-395. SPI/I2S firmware function

Function name	Function description
spi_i2s_deinit	reset SPI and I2S peripheral
spi_struct_para_init	initialize the parameters of SPI struct with the default values
spi_init	initialize SPI peripheral parameter
spi_enable	enable SPI
spi_disable	disable SPI
i2s_init	initialize I2S peripheral parameter
i2s_psc_config	configure I2S peripheral prescaler
i2s_enable	enable I2S
i2s_disable	disable I2S
spi_nss_output_enable	enable SPI NSS output function
spi_nss_output_disable	disable SPI NSS output function
spi_nss_internal_high	SPI NSS pin high level in software mode
spi_nss_internal_low	SPI NSS pin low level in software mode
spi_dma_enable	enable SPI DMA function
spi_dma_disable	disable SPI DMA function
spi_i2s_data_frame_format_config	configure SPI/I2S data frame format
spi_i2s_data_transmit	SPI transmit data
spi_i2s_data_receive	SPI receive data
spi_bidirectional_transfer_config	configure SPI bidirectional transfer direction
spi_crc_polynomial_set	set SPI CRC polynomial
spi_crc_polynomial_get	get SPI CRC polynomial
spi_crc_on	turn on SPI CRC function
spi_crc_off	turn off SPI CRC function
spi_crc_next	SPI next data is CRC value
spi_crc_get	get SPI CRC send value or receive value
spi_ti_mode_enable	enable SPI TI mode
spi_ti_mode_disable	disable SPI TI mode
spi_nssp_mode_enable	enable SPI NSS pulse mode
spi_nssp_mode_disable	disable SPI NSS pulse mode
spi_quad_enable	enable quad wire SPI
spi_quad_disable	disable quad wire SPI
spi_quad_write_enable	enable quad wire SPI write
spi_quad_read_enable	enable quad wire SPI read
spi_i2s_interrupt_enable	enable SPI and I2S interrupt
spi_i2s_interrupt_disable	disable SPI and I2S interrupt
spi_i2s_interrupt_flag_get	get SPI and I2S interrupt status
spi_i2s_flag_get	get SPI and I2S flag status
spi_crc_error_clear	clear SPI CRC error flag status

Structure spi_parameter_struct

Table 3-396. spi_parameter_struct

Member name	Function description
device_mode	SPI master or slave (SPI_MASTER, SPI_SLAVE)
trans_mode	SPI transtype (SPI_TRANSMODE_FULLDUPLEX, SPI_TRANSMODE_RECEIVEONLY, SPI_TRANSMODE_BDRECEIVE, SPI_TRANSMODE_BDTRANSMIT)
frame_size	SPI frame size (SPI_FRAMESIZE_16BIT, SPI_FRAMESIZE_8BIT)
nss	SPI NSS control by handware or software (SPI_NSS_SOFT, SPI_NSS_HARD)
endian	SPI big endian or little endian (SPI_ENDIAN_MSB, SPI_ENDIAN_LSB)
clock_polarity_phase	SPI clock phase and polarity (SPI_CK_PL_LOW_PH_1EDGE, SPI_CK_PL_HIGH_PH_1EDGE, SPI_CK_PL_LOW_PH_2EDGE, SPI_CK_PL_HIGH_PH_2EDGE)
prescale	SPI prescale factor (SPI_PSC_n (n=2,4,8,16,32,64,128,256))

spi_i2s_deinit

The description of spi_i2s_deinit is shown as below:

Table 3-397. Function spi_i2s_deinit

Function name	spi_i2s_deinit
Function prototype	void spi_i2s_deinit(uint32_t spi_periph);
Function descriptions	reset SPI and I2S peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
spi_periph	SPI/I2S peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset SPI0 */

spi_i2s_deinit(SPI0);
```

spi_struct_para_init

The description of spi_struct_para_init is shown as below:

Table 3-398. Function spi_struct_para_init

Function name	spi_struct_para_init
Function prototype	void spi_struct_para_init(spi_parameter_struct* spi_struct);
Function descriptions	initialize the parameters of SPI struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
*spi_struct	a spi_parameter_struct address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of SPI */
spi_parameter_struct spi_init_struct;
spi_struct_para_init(&spi_init_struct);
```

spi_init

The description of spi_init is shown as below:

Table 3-399. Function spi_init

Function name	spi_init
Function prototype	void spi_init(uint32_t spi_periph, spi_parameter_struct* spi_struct);
Function descriptions	initialize SPI peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
spi_struct	SPI parameter initialization stuct, the structure members can refer to members of the structure *Table 3-396. spi_parameter_struct*
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize SPI0 */
spi_parameter_struct spi_init_struct;
spi_init_struct.trans_mode           = SPI_TRANSMODE_BDTRANSMIT;
spi_init_struct.device_mode          = SPI_MASTER;
spi_init_struct.frame_size           = SPI_FRAMESIZE_8BIT;
spi_init_struct.clock_polarity_phase = SPI_CK_PL_HIGH_PH_2EDGE;
spi_init_struct.nss                  = SPI_NSS_SOFT;
spi_init_struct.prescale             = SPI_PSC_8;
spi_init_struct.endian               = SPI_ENDIAN_MSB;
spi_init(SPI0, &spi_init_struct);
```

spi_enable

The description of spi_enable is shown as below:

Table 3-400. Function spi_enable

Function name	spi_enable
Function prototype	void spi_enable(uint32_t spi_periph);
Function descriptions	enable SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 */
spi_enable(SPI0);
```

spi_disable

The description of spi_disable is shown as below:

Table 3-401. Function spi_disable

Function name	spi_disable
Function prototype	void spi_disable(uint32_t spi_periph);

Function descriptions	disable SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 */
spi_disable(SPI0);
```

i2s_init

The description of i2s_init is shown as below:

Table 3-402. Function i2s_init

Function name	i2s_init
Function prototype	void i2s_init(uint32_t spi_periph,uint32_t mode, uint32_t standard, uint32_t ckpl);
Function descriptions	initialize I2S peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=1,2
Input parameter{in}	
mode	I2S operation mode
I2S_MODE_SLAVETX	I2S slave transmit mode
I2S_MODE_SLAVERX	I2S slave receive mode
I2S_MODE_MASTERTX	I2S master transmit mode
I2S_MODE_MASTERRX	I2S master receive mode
Input parameter{in}	
standard	I2S standard
I2S_STD_PHILIPS	I2S philips standard
I2S_STD_MSB	I2S MSB standard
I2S_STD_LSB	I2S LSB standard
I2S_STD_PCMSHORT	I2S PCM short standard

I2S_STD_PCMLONG	I2S PCM long standard
Input parameter{in}	
ckpl	I2S idle state clock polarity
I2S_CKPL_LOW	I2S clock polarity low level
I2S_CKPL_HIGH	I2S clock polarity high level
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize I2S1 */
i2s_init(SPI1, I2S_MODE_MASTERTX, I2S_STD_PHILIPS, I2S_CKPL_LOW);
```

i2s_psc_config

The description of i2s_psc_config is shown as below:

Table 3-403. Function i2s_psc_config

Function name	i2s_psc_config
Function prototype	void i2s_psc_config(uint32_t spi_periph, uint32_t audiosample, uint32_t frameformat, uint32_t mckout);
Function descriptions	configure I2S prescaler
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=1,2
Input parameter{in}	
audiosample	I2S audio sample rate
I2S_AUDIOSAMPLE_8K	audio sample rate is 8KHz
I2S_AUDIOSAMPLE_11K	audio sample rate is 11KHz
I2S_AUDIOSAMPLE_16K	audio sample rate is 16KHz
I2S_AUDIOSAMPLE_22K	audio sample rate is 22KHz
I2S_AUDIOSAMPLE_32K	audio sample rate is 32KHz
I2S_AUDIOSAMPLE_44K	audio sample rate is 44KHz
I2S_AUDIOSAMPLE_4	audio sample rate is 48KHz

8K	
I2S_AUDIOSAMPLE_96K	audio sample rate is 96KHz
I2S_AUDIOSAMPLE_192K	audio sample rate is 192KHz
Input parameter{in}	
frameformat	I2S data length and channel length
I2S_FRAMEFORMAT_DT16B_CH16B	I2S data length is 16 bit and channel length is 16 bit
I2S_FRAMEFORMAT_DT16B_CH32B	I2S data length is 16 bit and channel length is 32 bit
I2S_FRAMEFORMAT_DT24B_CH32B	I2S data length is 24 bit and channel length is 32 bit
I2S_FRAMEFORMAT_DT32B_CH32B	I2S data length is 32 bit and channel length is 32 bit
Input parameter{in}	
mckout	I2S master clock output
I2S_MCKOUT_ENABLE	I2S master clock output enable
I2S_MCKOUT_DISABLE	I2S master clock output disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure I2S1 prescaler */

i2s_psc_config(SPI1, I2S_AUDIOSAMPLE_44K, I2S_FRAMEFORMAT_DT16B_CH16B, I2S_MCKOUT_DISABLE);

i2s_enable

The description of i2s_enable is shown as below:

Table 3-404. Function i2s_enable

Function name	i2s_enable
Function prototype	void i2s_enable(uint32_t spi_periph);
Function descriptions	enable I2S
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral

SPIx	x=1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable I2S1*/

i2s_enable(SPI1);

i2s_disable

The description of i2s_disable is shown as below:

Table 3-405. Function i2s_disable

Function name	i2s_disable
Function prototype	void i2s_disable(uint32_t spi_periph);
Function descriptions	disable I2S
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable I2S1*/

i2s_disable(SPI1);

spi_nss_output_enable

The description of spi_nss_output_enable is shown as below:

Table 3-406. Function spi_nss_output_enable

Function name	spi_nss_output_enable
Function prototype	void spi_nss_output_enable(uint32_t spi_periph);
Function descriptions	enable SPI NSS output function
Precondition	-
The called functions	-
Input parameter{in}	

spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 NSS output */

spi_nss_output_enable(SPI0);
```

spi_nss_output_disable

The description of spi_nss_output_disable is shown as below:

Table 3-407. Function spi_nss_output_disable

Function name	spi_nss_output_disable
Function prototype	void spi_nss_output_disable(uint32_t spi_periph);
Function descriptions	disable SPI NSS output function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 NSS output */

spi_nss_output_disable(SPI0);
```

spi_nss_internal_high

The description of spi_nss_internal_high is shown as below:

Table 3-408. Function spi_nss_internal_high

Function name	spi_nss_internal_high
Function prototype	void spi_nss_internal_high(uint32_t spi_periph);
Function descriptions	SPI NSS pin high level in software mode
Precondition	-
The called functions	-

Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 NSS pin is pulled high level in software mode */
spi_nss_internal_high(SPI0);
```

spi_nss_internal_low

The description of spi_nss_internal_low is shown as below:

Table 3-409. Function spi_nss_internal_low

Function name	spi_nss_internal_low
Function prototype	void spi_nss_internal_low(uint32_t spi_periph);
Function descriptions	SPI NSS pin low level in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 NSS pin is pulled low level in software mode */
spi_nss_internal_low(SPI0);
```

spi_dma_enable

The description of spi_dma_enable is shown as below:

Table 3-410. Function spi_dma_enable

Function name	spi_dma_enable
Function prototype	void spi_dma_enable(uint32_t spi_periph, uint8_t dma);
Function descriptions	enable SPI DMA function
Precondition	-

The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
dma	SPI DMA mode
SPI_DMA_TRANSMIT	SPI transmit data use DMA
SPI_DMA_RECEIVE	SPI receive data use DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 transmit data DMA function */
spi_dma_enable(SPI0, SPI_DMA_TRANSMIT);
```

spi_dma_disable

The description of spi_dma_disable is shown as below:

Table 3-411. Function spi_dma_disable

Function name	spi_dma_disable
Function prototype	void spi_dma_disable(uint32_t spi_periph, uint8_t dma);
Function descriptions	disable SPI DMA function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
dma	SPI DMA mode
SPI_DMA_TRANSMIT	SPI transmit data use DMA
SPI_DMA_RECEIVE	SPI receive data use DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 transmit data DMA function */
spi_dma_disable(SPI0, SPI_DMA_TRANSMIT);
```

spi_i2s_data_frame_format_config

The description of spi_i2s_data_frame_format_config is shown as below:

Table 3-412. Function spi_i2s_data_frame_format_config

Function name	spi_i2s_data_frame_format_config
Function prototype	void spi_i2s_data_frame_format_config(uint32_t spi_periph, uint16_t frame_format);
Function descriptions	configure SPI/I2S data frame format
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
frame_format	SPI frame size
SPI_FRAMESIZE_16BIT	SPI frame size is 16 bits
SPI_FRAMESIZE_8BIT	SPI frame size is 8 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure SPI1/I2S1 data frame format size is 16 bits */

spi_i2s_data_frame_format_config(SPI1, SPI_FRAMESIZE_16BIT);

spi_i2s_data_transmit

The description of spi_i2s_data_transmit is shown as below:

Table 3-413. Function spi_i2s_data_transmit

Function name	spi_i2s_data_transmit
Function prototype	void spi_i2s_data_transmit(uint32_t spi_periph, uint16_t data);
Function descriptions	SPI transmit data
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
data	16-bit data
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* SPI0 transmit data */

spi_i2s_data_transmit(SPI0, spi0_send_array[send_n]);

spi_i2s_data_receive

The description of spi_i2s_data_receive is shown as below:

Table 3-414. Function spi_i2s_data_receive

Function name	spi_i2s_data_receive
Function prototype	uint16_t spi_i2s_data_receive(uint32_t spi_periph);
Function descriptions	SPI receive data
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit data

Example:

/* SPI0 receive data */

spi0_receive_array[receive_n] = spi_i2s_data_receive(SPI0);

spi_bidirectional_transfer_config

The description of spi_bidirectional_transfer_config is shown as below:

Table 3-415. Function spi_bidirectional_transfer_config

Function name	spi_bidirectional_transfer_config
Function prototype	void spi_bidirectional_transfer_config(uint32_t spi_periph, uint32_t transfer_direction);
Function descriptions	configure SPI bidirectional transfer direction
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral

SPIx	x=0,1,2
Input parameter{in}	
transfer_direction	SPI transfer direction
SPI_BIDIRECTIONAL_TRANSMIT	SPI work in transmit-only mode
SPI_BIDIRECTIONAL_RECEIVE	SPI work in receive-only mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* SPI0 works in transmit-only mode */

spi_bidirectional_transfer_config(SPI0, SPI_BIDIRECTIONAL_TRANSMIT);

spi_crc_polynomial_set

The description of spi_crc_polynomial_set is shown as below:

Table 3-416. Function spi_crc_polynomial_set

Function name	spi_crc_polynomial_set
Function prototype	void spi_crc_polynomial_set(uint32_t spi_periph, uint16_t crc_poly);
Function descriptions	set SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
crc_poly	CRC polynomial value
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* set SPI0 CRC polynomial */

spi_crc_polynomial_set(SPI0,CRC_VALUE);

spi_crc_polynomial_get

The description of spi_crc_polynomial_get is shown as below:

Table 3-417. Function spi_crc_polynomial_get

Function name	spi_crc_polynomial_get
Function prototype	uint16_t spi_crc_polynomial_get(uint32_t spi_periph);
Function descriptions	get SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit CRC polynomial (0-0xFFFF)

Example:

```
/* get SPI0 CRC polynomial */
uint16_t crc_val;
crc_val = spi_crc_polynomial_get(SPI0);
```

spi_crc_on

The description of spi_crc_on is shown as below:

Table 3-418. Function spi_crc_on

Function name	spi_crc_on
Function prototype	void spi_crc_on(uint32_t spi_periph);
Function descriptions	turn on SPI CRC function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn on SPI0 CRC function */
spi_crc_on(SPI0);
```

spi_crc_off

The description of spi_crc_off is shown as below:

Table 3-419. Function spi_crc_off

Function name	spi_crc_off
Function prototype	void spi_crc_off(uint32_t spi_periph);
Function descriptions	turn off SPI CRC function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn off SPI0 CRC function */
spi_crc_off(SPI0);
```

spi_crc_next

The description of spi_crc_next is shown as below:

Table 3-420. Function spi_crc_next

Function name	spi_crc_next
Function prototype	void spi_crc_next(uint32_t spi_periph);
Function descriptions	SPI next data is CRC value
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 next data is CRC value */
spi_crc_next(SPI0);
```

spi_crc_get

The description of spi_crc_get is shown as below:

Table 3-421. Function spi_crc_get

Function name	spi_crc_get
Function prototype	uint16_t spi_crc_get(uint32_t spi_periph, uint8_t crc);
Function descriptions	get SPI CRC send value or receive value
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
crc	SPI crc value
SPI_CRC_TX	get transmit crc value
SPI_CRC_RX	get receive crc value
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit CRC value (0-0xFFFF)

Example:

```
/* get SPI0 CRC send value */
uint16_t crc_val;
crc_val = spi_crc_get(SPI0, SPI_CRC_TX);
```

spi_ti_mode_enable

The description of spi_ti_mode_enable is shown as below:

Table 3-422. Function spi_ti_mode_enable

Function name	spi_ti_mode_enable
Function prototype	void spi_ti_mode_enable(uint32_t spi_periph);
Function descriptions	enable SPI TI mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* enable SPI0 TI mode */

spi_ti_mode_enable(SPI0);

spi_ti_mode_disable

The description of spi_ti_mode_disable is shown as below:

Table 3-423. Function spi_ti_mode_disable

Function name	spi_ti_mode_disable
Function prototype	void spi_ti_mode_disable(uint32_t spi_periph);
Function descriptions	disable SPI TI mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable SPI0 TI mode */

spi_ti_mode_disable(SPI0);

spi_nssp_mode_enable

The description of spi_nssp_mode_enable is shown as below:

Table 3-424. Function spi_nssp_mode_enable

Function name	spi_nssp_mode_enable
Function prototype	void spi_nssp_mode_enable(uint32_t spi_periph);
Function descriptions	enable SPI NSS pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* enable SPI0 NSS pulse mode */

spi_nssp_mode_enable(SPI0);

spi_nssp_mode_disable

The description of spi_nssp_mode_disable is shown as below:

Table 3-425. Function spi_nssp_mode_disable

Function name	spi_nssp_mode_disable
Function prototype	void spi_nssp_mode_disable(uint32_t spi_periph);
Function descriptions	disable SPI NSS pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable SPI0 NSS pulse mode */

spi_nssp_mode_disable(SPI0);

spi_quad_enable

The description of spi_quad_enable is shown as below:

Table 3-426. Function spi_quad_enable

Function name	spi_quad_enable
Function prototype	void spi_quad_enable(uint32_t spi_periph);
Function descriptions	enable quad wire SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* enable SPI0 quad wire mode */

spi_quad_enable(SPI0);

spi_quad_disable

The description of spi_quad_disable is shown as below:

Table 3-427. Function spi_quad_disable

Function name	spi_quad_disable
Function prototype	spi_quad_disable(uint32_t spi_periph);
Function descriptions	disable quad wire SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable SPI0 quad wire mode */

spi_quad_disable(SPI0);

spi_quad_write_enable

The description of spi_quad_write_enable is shown as below:

Table 3-428. Function spi_quad_write_enable

Function name	spi_quad_write_enable
Function prototype	void spi_quad_write_enable(uint32_t spi_periph);
Function descriptions	enable quad wire SPI write
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 quad wire write */
spi_quad_write_enable(SPI0);
```

spi_quad_read_enable

The description of spi_quad_read_enable is shown as below:

Table 3-429. Function spi_quad_read_enable

Function name	spi_quad_read_enable
Function prototype	void spi_quad_read_enable(uint32_t spi_periph);
Function descriptions	enable quad wire SPI read
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 quad wire read */
spi_quad_read_enable(SPI0);
```

spi_i2s_interrupt_enable

The description of spi_i2s_interrupt_enable is shown as below:

Table 3-430. Function spi_i2s_interrupt_enable

Function name	spi_i2s_interrupt_enable
Function prototype	void spi_i2s_interrupt_enable(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	enable SPI and I2S interrupt
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral

SPIx	x=0,1,2
Input parameter{in}	
interrupt	SPI/I2S interrupt
SPI_I2S_INT_TBE	transmit buffer empty interrupt
SPI_I2S_INT_RBNE	receive buffer not empty interrupt
SPI_I2S_INT_ERR	CRC error,configuration error,reception overrun error, transmission underrun error and format error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 transmit buffer empty interrupt */
spi_i2s_interrupt_enable(SPI0, SPI_I2S_INT_TBE);
```

spi_i2s_interrupt_disable

The description of spi_i2s_interrupt_disable is shown as below:

Table 3-431. Function spi_i2s_interrupt_disable

Function name	spi_i2s_interrupt_disable
Function prototype	void spi_i2s_interrupt_disable(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	disable SPI and I2S interrupt
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
interrupt	SPI/I2S interrupt
SPI_I2S_INT_TBE	transmit buffer empty interrupt
SPI_I2S_INT_RBNE	receive buffer not empty interrupt
SPI_I2S_INT_ERR	CRC error,configuration error,reception overrun error, transmission underrun error and format error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 transmit buffer empty interrupt */
spi_i2s_interrupt_disable(SPI0, SPI_I2S_INT_TBE);
```

spi_i2s_interrupt_flag_get

The description of spi_i2s_interrupt_flag_get is shown as below:

Table 3-432. Function spi_i2s_interrupt_flag_get

Function name	spi_i2s_interrupt_flag_get
Function prototype	FlagStatus spi_i2s_interrupt_flag_get(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	get SPI and I2S interrupt status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
interrupt	SPI/I2S interrupt flag status
SPI_I2S_INT_FLAG_TBE	transmit buffer empty interrupt
SPI_I2S_INT_FLAG_RBNE	receive buffer not empty interrupt
SPI_I2S_INT_FLAG_RXORERR	overrun interrupt
SPI_INT_FLAG_CONFERR	config error interrupt
SPI_INT_FLAG_CRCERR	CRC error interrupt
I2S_INT_FLAG_TXURERR	underrun error interrupt
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SPI0 transmit buffer empty interrupt status */
if(RESET != spi_i2s_interrupt_flag_get(SPI0, SPI_I2S_INT_FLAG_TBE)){
    while(RESET == spi_i2s_flag_get(SPI0, SPI_FLAG_TBE));
    spi_i2s_data_transmit(SPI0, spi0_send_array[send_n++]);
}
```

spi_i2s_flag_get

The description of spi_i2s_flag_get is shown as below:

Table 3-433. Function spi_i2s_flag_get

Function name	spi_i2s_flag_get
Function prototype	FlagStatus spi_i2s_flag_get(uint32_t spi_periph, uint32_t flag);
Function descriptions	get SPI and I2S flag status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
flag	SPI/I2S flag status
SPI_FLAG_TBE	transmit buffer empty flag
SPI_FLAG_RBNE	receive buffer not empty flag
SPI_FLAG_TRANS	transmit on-going flag
SPI_I2S_INT_FLAG_RXORERR	receive overrun error flag
SPI_FLAG_CONFERR	mode config error flag
SPI_FLAG_CRCERR	CRC error flag
I2S_FLAG_RXORERR	overrun error flag
I2S_FLAG_TXURERR	underrun error flag
I2S_FLAG_CH	channel side flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SPI0 transmit buffer empty flag status */
while(RESET == spi_i2s_flag_get(SPI0, SPI_FLAG_TBE));
spi_i2s_data_transmit(SPI0, spi0_send_array[send_n++]);
```

spi_crc_error_clear

The description of spi_crc_error_clear is shown as below:

Table 3-434. Function spi_crc_error_clear

Function name	spi_crc_error_clear
Function prototype	void spi_crc_error_clear(uint32_t spi_periph);
Function descriptions	clear SPI CRC error flag status
Precondition	-
The called functions	-
Input parameter{in}	

spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear SPI0 CRC error flag status */
spi_crc_error_clear(SPI0);
```

3.20. TIMER

The timers have a 16-bit counter that can be used as an unsigned counter and supports both input capture and output compare. Timers (TIMERx) are divided into five sorts: advanced timer (TIMERx, x=0, 7), general level0 timer (TIMERx, x=1, 2, 3, 4), general level1 timer (TIMERx, x=8, 11), general level2 timer (TIMERx, x=9, 10, 12, 13), Basic timer (TIMERx, x=5, 6). The specific functions of different types of timer are different. The TIMER registers are listed in chapter *3.20.1*, the TIMER firmware functions are introduced in chapter *3.20.2.*

3.20.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

Table 3-435. TIMERx Registers

Registers	Descriptions
TIMER_CTL0	Control register 0
TIMER_CTL1	Control register 1
TIMER_SMCFG	Slave mode configuration register
TIMER_DMAINTEN	DMA and interrupt enable register
TIMER_INTF	Interrupt flag register
TIMER_SWEVG	Software event generation register
TIMER_CHCTL0	Channel control register 0
TIMER_CHCTL1	Channel control register 1
TIMER_CHCTL2	Channel control register 2
TIMER_CNT	Counter register
TIMER_PSC	Prescaler register
TIMER_CAR	Counter auto reload register
TIMER_CREP	Counter repetition register
TIMER_CH0CV	Channel 0 capture/compare value register
TIMER_CH1CV	Channel 1 capture/compare value register
TIMER_CH2CV	Channel 2 capture/compare value register

Registers	Descriptions
TIMER_CH3CV	Channel 3 capture/compare value register
TIMER_CCHP	Channel complementary protection register
TIMER_DMACFG	DMA configuration register
TIMER_DMATB	DMA transfer buffer register
TIMER_CFG	Configuration register

3.20.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

Table 3-436. TIMERx firmware function

Function name	Function description
timer_deinit	deinit a timer
timer_struct_para_init	initialize the parameters of TIMER init parameter struct with the default values
timer_init	initialize TIMER counter
timer_enable	enable a timer
timer_disable	disable a timer
timer_auto_reload_shadow_enable	enable the auto reload shadow function
timer_auto_reload_shadow_disable	disable the auto reload shadow function
timer_update_event_enable	enable the update event
timer_update_event_disable	disable the update event
timer_counter_alignment	set TIMER counter alignment mode
timer_counter_up_direction	set TIMER counter up direction
timer_counter_down_direction	set TIMER counter down direction
timer_prescaler_config	configure TIMER prescaler
timer_repetition_value_config	configure TIMER repetition register value
timer_autoreload_value_config	configure TIMER autoreload register value
timer_counter_value_config	configure TIMER counter register value
timer_counter_read	read TIMER counter value
timer_prescaler_read	read TIMER prescaler value
timer_single_pulse_mode_config	configure TIMER single pulse mode
timer_update_source_config	configure TIMER update source
timer_dma_enable	enable the TIMER DMA
timer_dma_disable	disable the TIMER DMA
timer_channel_dma_request_source_select	channel DMA request source selection
timer_dma_transfer_config	configure the TIMER DMA transfer
timer_event_software_generate	software generate events
timer_break_struct_para_init	initialize the parameters of TIMER break parameter struct with the default values
timer_break_config	configure TIMER break function

Function name	Function description
timer_break_enable	enable TIMER break function
timer_break_disable	disable TIMER break function
timer_automatic_output_enable	enable TIMER output automatic function
timer_automatic_output_disable	disable TIMER output automatic function
timer_primary_output_config	configure TIMER primary output function
timer_channel_control_shadow_ config	channel capture/compare control shadow register enable
timer_channel_control_shadow_ update_config	configure TIMER channel control shadow register update control
timer_channel_output_struct _para_init	initialize the parameters of TIMER channel output parameter struct with the default values
timer_channel_output_config	configure TIMER channel output function
timer_channel_output_mode_config	configure TIMER channel output compare mode
timer_channel_output_pulse_ value_config	configure TIMER channel output pulse value
timer_channel_output_shadow_ config	configure TIMER channel output shadow function
timer_channel_output_fast_config	configure TIMER channel output fast function
timer_channel_output_clear_config	configure TIMER channel output clear function
timer_channel_output_polarity_ config	configure TIMER channel output polarity
timer_channel_complementary_ output_polarity_config	configure TIMER channel complementary output polarity
timer_channel_output_state_config	configure TIMER channel enable state
timer_channel_complementary_ output_state_config	configure TIMER channel complementary output enable state
timer_channel_input_struct_ para_init	initialize the parameters of TIMER channel input parameter struct with the default values
timer_input_capture_config	configure TIMER input capture parameter
timer_channel_input_capture_ prescaler_config	configure TIMER channel input capture prescaler value
timer_channel_capture_value_ register_read	read TIMER channel capture compare register valu
timer_input_pwm_capture_config	configure TIMER input pwm capture function
timer_hall_mode_config	configure TIMER hall sensor mode
timer_input_trigger_source_select	select TIMER input trigger source
timer_master_output_trigger_ source_select	select TIMER master mode output trigger source
timer_slave_mode_select	select TIMER slave mode
timer_master_slave_mode_config	configure TIMER master slave mode
timer_external_trigger_config	configure TIMER external trigger input

Function name	Function description
timer_quadrature_decoder_mode_config	configure TIMER quadrature decoder mode
timer_internal_clock_config	configure TIMER internal clock mode
timer_internal_trigger_as_external_clock_config	configure TIMER the internal trigger as external clock input
timer_external_trigger_as_external_clock_config	configure TIMER the external trigger as external clock input
timer_external_clock_mode0_config	configure TIMER the external clock mode 0
timer_external_clock_mode1_config	configure TIMER the external clock mode 1
timer_external_clock_mode1_disable	disable TIMER the external clock mode 1
timer_write_chxval_register_config	configure TIMER write CHxVAL register selection
timer_output_value_selection_config	configure TIMER output value selection
timer_interrupt_enable	enable the TIMER interrupt
timer_interrupt_disable	disable the TIMER interrupt
timer_interrupt_flag_get	get timer interrupt flag
timer_interrupt_flag_clear	clear TIMER interrupt flag
timer_flag_get	get TIMER flags
timer_flag_clear	clear TIMER flags

Structure timer_parameter_struct

Table 3-437. Structure timer_parameter_struct

Member name	Function description
prescaler	prescaler value(0~65535)
alignedmode	aligned mode(TIMER_COUNTER_EDGE, TIMER_COUNTER_CENTER_DOWN, TIMER_COUNTER_CENTER_UP, TIMER_COUNTER_CENTER_BOTH)
counterdirection	counter direction(TIMER_COUNTER_UP, TIMER_COUNTER_DOWN)
period	period value(0~65535)
clockdivision	clock division value(TIMER_CKDIV_DIV1, TIMER_CKDIV_DIV2, TIMER_CKDIV_DIV4)
repetitioncounter	the counter repetition value(0~255)

Structure timer_break_parameter_struct

Table 3-438. Structure timer_break_parameter_struct

Member name	Function description
runoffstate	run mode off-state(TIMER_ROS_STATE_ENABLE, TIMER_ROS_STATE_DISABLE)
ideloffstate	idle mode off-state(TIMER_IOS_STATE_ENABLE,

Member name	Function description
	TIMER_IOS_STATE_DISABLE)
deadtime	dead time(0~255)
breakpolarity	break polarity(TIMER_BREAK_POLARITY_LOW, TIMER_BREAK_POLARITY_HIGH)
outputautostate	output automatic enable (TIMER_OUTAUTO_ENABLE, TIMER_OUTAUTO_DISABLE)
protectmode	complementary register protect control(TIMER_CCHP_PROT_OFF, TIMER_CCHP_PROT_0, TIMER_CCHP_PROT_1, TIMER_CCHP_PROT_2)
breakstate	break enable(TIMER_BREAK_ENABLE, TIMER_BREAK_DISABLE)

Structure timer_oc_parameter_struct

Table 3-439. Structure timer_oc_parameter_struct

Member name	Function description
outputstate	channel output state(TIMER_CCX_ENABLE, TIMER_CCX_DISABLE)
outputnstate	channel complementary output state(TIMER_CCXN_ENABLE, TIMER_CCXN_DISABLE)
ocpolarity	channel output polarity(TIMER_OC_POLARITY_HIGH, TIMER_OC_POLARITY_LOW)
ocnpolarity	channel complementary output polarity(TIMER_OCN_POLARITY_HIGH, TIMER_OCN_POLARITY_LOW)
ocidlestate	idle state of channel output(TIMER_OC_IDLE_STATE_LOW, TIMER_OC_IDLE_STATE_HIGH)
ocnidlestate	idle state of channel complementary output(TIMER_OCN_IDLE_STATE_LOW, TIMER_OCN_IDLE_STATE_HIGH)

Structure timer_ic_parameter_struct

Table 3-440. Structure timer_ic_parameter_struct

Member name	Function description
icpolarity	channel input polarity(TIMER_IC_POLARITY_RISING, TIMER_IC_POLARITY_FALLING, TIMER_IC_POLARITY_BOTH_EDGE)
icselection	channel input mode selection(TIMER_IC_SELECTION_DIRECTTI, TIMER_IC_SELECTION_INDIRECTTI, TIMER_IC_SELECTION_ITS)
icprescaler	channel input capture prescaler(TIMER_IC_PSC_DIV1, TIMER_IC_PSC_DIV2, TIMER_IC_PSC_DIV4, TIMER_IC_PSC_DIV8)
icfilter	channel input capture filter control(0~15)

timer_deinit

The description of timer_deinit is shown as below:

Table 3-441. Function timer_deinit

Function name	timer_deinit
Function prototype	void timer_deinit(uint32_t timer_periph);
Function descriptions	deinit a TIMER
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset TIMER0 */
timer_deinit (TIMER0);
```

timer_struct_para_init

The description of timer_struct_para_init is shown as below:

Table 3-442. Function timer_struct_para_init

Function name	timer_struct_para_init
Function prototype	void timer_struct_para_init(timer_parameter_struct* initpara);
Function descriptions	initialize the parameters of TIMER init parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to ***Table 3-437. Structure timer_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER init parameter struct with a default value */
timer_parameter_struct timer_initpara;
timer_struct_para_init(timer_initpara);
```

timer_init

The description of timer_init is shown as below:

Table 3-443. Function timer_init

Function name	timer_init
Function prototype	void timer_init(uint32_t timer_periph, timer_parameter_struct* initpara);
Function descriptions	initialize TIMER counter
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to ***Table 3-437. Structure timer_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER0 */
timer_parameter_struct timer_initpara;
timer_initpara.prescaler         = 107;
timer_initpara.alignedmode       = TIMER_COUNTER_EDGE;
timer_initpara.counterdirection  = TIMER_COUNTER_UP;
timer_initpara.period            = 999;
timer_initpara.clockdivision     = TIMER_CKDIV_DIV1;
timer_initpara.repetitioncounter = 1;
timer_init(TIMER0,&timer_initpara);
```

timer_enable

The description of timer_enable is shown as below:

Table 3-444. Function timer_enable

Function name	timer_enable
Function prototype	void timer_enable(uint32_t timer_periph);
Function descriptions	enable a timer

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 */
timer_enable (TIMER0);
```

timer_disable

The description of timer_disable is shown as below:

Table 3-445. Function timer_disable

Function name	timer_disable
Function prototype	void timer_disable(uint32_t timer_periph);
Function descriptions	disable a timer
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 */
timer_disable (TIMER0);
```

timer_auto_reload_shadow_enable

The description of timer_auto_reload_shadow_enable is shown as below:

Table 3-446. Function timer_auto_reload_shadow_enable

Function name	timer_auto_reload_shadow_enable
Function prototype	void timer_auto_reload_shadow_enable(uint32_t timer_periph);

Function descriptions	enable the auto reload shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMER0 auto reload shadow function */
timer_auto_reload_shadow_enable (TIMER0);
```

timer_auto_reload_shadow_disable

The description of timer_auto_reload_shadow_disable is shown as below:

Table 3-447. Function timer_auto_reload_shadow_disable

Function name	timer_auto_reload_shadow_ disable
Function prototype	void timer_auto_reload_shadow_ disable (uint32_t timer_periph);
Function descriptions	disable the auto reload shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMER0 auto reload shadow function */
timer_auto_reload_shadow_disable (TIMER0);
```

timer_update_event_enable

The description of timer_update_event_enable is shown as below:

Table 3-448. Function timer_update_event_enable

Function name	timer_update_event_enable

Function prototype	void timer_update_event_enable(uint32_t timer_periph);
Function descriptions	enable the update event
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable TIMER0 the update event */

timer_update_event_enable (TIMER0);

timer_update_event_disable

The description of timer_update_event_disable is shown as below:

Table 3-449. Function timer_update_event_disable

Function name	timer_update_event_ disable
Function prototype	void timer_update_event_ disable (uint32_t timer_periph);
Function descriptions	disable the update event
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TIMER0 the update event */

timer_update_event_disable (TIMER0);

timer_counter_alignment

The description of timer_counter_alignment is shown as below:

Table 3-450. Function timer_counter_alignment

Function name	timer_counter_alignment
Function prototype	void timer_counter_alignment(uint32_t timer_periph, uint16_t aligned);
Function descriptions	set TIMER counter alignment mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7..13)	TIMER peripheral selection
Input parameter{in}	
aligned	alignment mode
TIMER_COUNTER_EDGE	No center-aligned mode (edge-aligned mode). The direction of the counter isspecified by the DIR bit.
TIMER_COUNTER_CENTER_DOWN	Center-aligned and counting down assert mode. The counter counts under center aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0register). Only when the counter is counting down, compare interrupt flag of channels can be set.
TIMER_COUNTER_CENTER_UP	Center-aligned and counting up assert mode. The counter counts under center aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0register). Only when the counter is counting up, compare interrupt flag of channels can be set.
TIMER_COUNTER_CENTER_BOTH	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 the counter is counting up and counting down, compare interrupt flag of channels can be set.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set TIMER0 counter center-aligned and counting up assert mode */
timer_counter_alignment (TIMER0, TIMER_COUNTER_CENTER_UP);
```

timer_counter_up_direction

The description of timer_counter_up_direction is shown as below:

Table 3-451. Function timer_counter_up_direction

Function name	timer_counter_up_direction
Function prototype	void timer_counter_up_direction(uint32_t timer_periph);
Function descriptions	set TIMER counter up direction
Precondition	set TIMER counter no center-aligned mode (edge-aligned mode)

The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set TIMER0 counter up direction */
timer_counter_up_direction (TIMER0);
```

timer_counter_down_direction

The description of timer_counter_down_direction is shown as below:

Table 3-452. timer_counter_down_direction

Function name	timer_counter_ down _direction
Function prototype	void timer_counter_ down _direction(uint32_t timer_periph);
Function descriptions	set TIMER counter down direction
Precondition	set TIMER counter no center-aligned mode (edge-aligned mode)
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set TIMER0 counter down direction */
timer_counter_down_direction (TIMER0);
```

timer_prescaler_config

The description of timer_prescaler_config is shown as below:

Table 3-453. Function timer_prescaler_config

Function name	timer_prescaler_config
Function prototype	void timer_prescaler_config(uint32_t timer_periph, uint16_t prescaler, uint8_t pscreload);

Function descriptions	configure TIMER prescaler
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Input parameter{in}	
prescaler	prescaler value (0~65535)
Input parameter{in}	
pscreload	prescaler reload mode
TIMER_PSC_RELOAD_NOW	the prescaler is loaded right now
TIMER_PSC_RELOAD_UPDATE	the prescaler is loaded at the next update event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 prescaler */
timer_prescaler_config (TIMER0, 3000, TIMER_PSC_RELOAD_NOW);
```

timer_repetition_value_config

The description of timer_repetition_value_config is shown as below:

Table 3-454. Function timer_repetition_value_config

Function name	timer_repetition_value_config
Function prototype	void timer_repetition_value_config(uint32_t timer_periph, uint16_t repetition);
Function descriptions	configure TIMER repetition register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
repetition	the counter repetition value (0~255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 repetition register value */

timer_repetition_value_config (TIMER0, 98);

timer_autoreload_value_config

The description of timer_autoreload_value_config is shown as below:

Table 3-455. Function timer_autoreload_value_config

Function name	timer_autoreload_value_config
Function prototype	void timer_autoreload_value_config(uint32_t timer_periph, uint16_t autoreload);
Function descriptions	configure TIMER autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Input parameter{in}	
autoreload	the counter auto-reload value (0-0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER autoreload register value */

timer_autoreload_value_config (TIMER0, 3000);

timer_counter_value_config

The description of timer_counter_value_config is shown as below:

Table 3-456. Function timer_counter_value_config

Function name	timer_counter_value_config
Function prototype	void timer_counter_value_config(uint32_t timer_periph, uint16_t counter);
Function descriptions	configure TIMER counter register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Input parameter{in}	

counter	the counter value (0-0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 counter register value */
timer_counter_value_config (TIMER0);
```

timer_counter_read

The description of timer_counter_read is shown as below:

Table 3-457. Function timer_counter_read

Function name	timer_counter_read
Function prototype	uint32_t timer_counter_read(uint32_t timer_periph);
Function descriptions	read TIMER counter value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint32_t	counter value(0x0000~0xFFFF)

Example:

```
/* read TIMER0 counter value */
uint32_t i = 0;
i = timer_counter_read (TIMER0);
```

timer_prescaler_read

The description of timer_prescaler_read is shown as below:

Table 3-458. Function timer_prescaler_read

Function name	timer_prescaler_read
Function prototype	uint16_t timer_prescaler_read(uint32_t timer_periph);
Function descriptions	read TIMER prescaler value
Precondition	-
The called functions	-

Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	prescaler register value (0x0000~0xFFFF)

Example:

```
/* read TIMER0 prescaler value */
uint16_t i = 0;
i = timer_prescaler_read (TIMER0);
```

timer_single_pulse_mode_config

The description of timer_single_pulse_mode_config is shown as below:

Table 3-459. Function timer_single_pulse_mode_config

Function name	timer_single_pulse_mode_config
Function prototype	void timer_single_pulse_mode_config(uint32_t timer_periph, uint8_t spmode);
Function descriptions	configure TIMER single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..8,11)	TIMER peripheral selection
Input parameter{in}	
spmode	pulse mode
TIMER_SP_MODE_SINGLE	single pulse mode
TIMER_SP_MODE_REPETITIVE	repetitive pulse mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 single pulse mode */
timer_single_pulse_mode_config (TIMER0, TIMER_SP_MODE_SINGLE);
```

timer_update_source_config

The description of timer_update_source_config is shown as below:

Table 3-460. Function timer_update_source_config

Function name	timer_update_source_config
Function prototype	void timer_update_source_config(uint32_t timer_periph, uint32_t update);
Function descriptions	configure TIMER update source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..13)	TIMER peripheral selection
Input parameter{in}	
update	update source
TIMER_UPDATE_SRC_GLOBAL	Any of the following events generate an update interrupt or DMA request: - The UPG bit is set - The counter generates an overflow or underflow event - The slave mode controller generates an update event
TIMER_UPDATE_SRC_REGULAR	Only counter overflow/underflow generates an update interrupt or DMA request.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER update only by counter overflow/underflow */
timer_update_source_config (TIMER0, TIMER_UPDATE_SRC_REGULAR);
```

timer_dma_enable

The description of timer_dma_enable is shown as below:

Table 3-461. Function timer_dma_enable

Function name	timer_dma_enable
Function prototype	void timer_dma_enable(uint32_t timer_periph, uint16_t dma);
Function descriptions	enable the TIMER DMA
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	

dma	timer DMA source enable
TIMER_DMA_UPD	update DMA enable, TIMERx(x=0..7)
TIMER_DMA_CH0D	channel 0 DMA enable, TIMERx(x=0..4,7)
TIMER_DMA_CH1D	channel 1 DMA enable, TIMERx(x=0..4,7)
TIMER_DMA_CH2D	channel 2 DMA enable, TIMERx(x=0..4,7)
TIMER_DMA_CH3D	channel 3 DMA enable, TIMERx(x=0..4,7)
TIMER_DMA_CMTD	commutation DMA request enable, TIMERx(x=0,7)
TIMER_DMA_TRGD	trigger DMA enable, TIMERx(x=0..4,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMER0 update DMA */
timer_dma_enable (TIMER0, TIMER_DMA_UPD);
```

timer_dma_disable

The description of timer_dma_disable is shown as below:

Table 3-462. Function timer_dma_disable

Function name	timer_dma_disable
Function prototype	void timer_dma_disable (uint32_t timer_periph, uint16_t dma);
Function descriptions	disable the TIMER DMA
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
dma	timer DMA source disable
TIMER_DMA_UPD	update DMA disable, TIMERx(x=0..7)
TIMER_DMA_CH0D	channel 0 DMA disable, TIMERx(x=0..4,7)
TIMER_DMA_CH1D	channel 1 DMA disable, TIMERx(x=0..4,7)
TIMER_DMA_CH2D	channel 2 DMA disable, TIMERx(x=0..4,7)
TIMER_DMA_CH3D	channel 3 DMA disable, TIMERx(x=0..4,7)
TIMER_DMA_CMTD	commutation DMA request disable, TIMERx(x=0,7)
TIMER_DMA_TRGD	trigger DMA disable, TIMERx(x=0..4,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable the TIMER0 update DMA */

timer_dma_disable (TIMER0, TIMER_DMA_UPD);

timer_channel_dma_request_source_select

The description of timer_channel_dma_request_source_select is shown as below:

Table 3-463. Function timer_channel_dma_request_source_select

Function name	timer_channel_dma_request_source_select
Function prototype	void timer_channel_dma_request_source_select(uint32_t timer_periph, uint32_t dma_request);
Function descriptions	channel DMA request source selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	TIMER peripheral selection
Input parameter{in}	
dma_request	channel DMA request source selection
TIMER_DMAREQUEST_CHANNELEVENT	DMA request of channel n is sent when channel y event occurs
TIMER_DMAREQUEST_UPDATEEVENT	DMA request of channel n is sent when update event occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* TIMER0 channel DMA request of channel n is sent when channel y event occurs */

timer_channel_dma_request_source_select(TIMER0,
TIMER_DMAREQUEST_CHANNELEVENT);

timer_dma_transfer_config

The description of timer_dma_transfer_config is shown as below:

Table 3-464. Function timer_dma_transfer_config

Function name	timer_dma_transfer_config
Function prototype	void timer_dma_transfer_config(uint32_t timer_periph, uint32_t dma_baseaddr, uint32_t dma_lenth);
Function descriptions	configure the TIMER DMA transfer

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
dma_baseaddr	DMA transfer access start address
TIMER_DMACFG_DMATA_CTL0	DMA transfer address is TIMER_CTL0, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CTL1	DMA transfer address is TIMER_CTL1, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_SMCFG	DMA transfer address is TIMER_SMCFG, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_DMAINTEN	DMA transfer address is TIMER_DMAINTEN, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_INTF	DMA transfer address is TIMER_INTF, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_SWEVG	DMA transfer address is TIMER_SWEVG, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CHCTL0	DMA transfer address is TIMER_CHCTL0, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CHCTL1	DMA transfer address is TIMER_CHCTL1, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CHCTL2	DMA transfer address is TIMER_CHCTL2, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CNT	DMA transfer address is TIMER_CNT, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_PSC	DMA transfer address is TIMER_PSC, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CAR	MA transfer address is TIMER_CAR, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CREP	DMA transfer address is TIMER_CREP, TIMERx(x=0,7)
TIMER_DMACFG_DMATA_CH0CV	DMA transfer address is TIMER_CH0CV, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CH1CV	DMA transfer address is TIMER_CH1CV, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CH2CV	DMA transfer address is TIMER_CH2CV, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CH3CV	DMA transfer address is TIMER_CH3CV, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_CCHP	DMA transfer address is TIMER_CCHP, TIMERx(x=0,7)

TIMER_DMACFG_DMATA_DMACFG	DMA transfer address is TIMER_DMACFG, TIMERx(x=0..4,7)
TIMER_DMACFG_DMATA_DMATB	DMA transfer address is TIMER_DMATB, TIMERx(x=0..4,7)
Input parameter{in}	
dma_lenth	DMA transfer count
TIMER_DMACFG_DMATC_xTRANSFER	x=1..18, DMA transfer x time
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the TIMER0 DMA transfer */

timer_dma_transfer_config(TIMER0, TIMER_DMACFG_DMATA_CTL0, TIMER_DMACFG_DMATC_5TRANSFER);

timer_event_software_generate

The description of timer_event_software_generate is shown as below:

Table 3-465. Function timer_event_software_generate

Function name	timer_event_software_generate
Function prototype	void timer_event_software_generate(uint32_t timer_periph, uint16_t event);
Function descriptions	software generate events
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
event	the timer software event generation sources
TIMER_EVENT_SRC_UPG	update event,TIMERx(x=0..13)
TIMER_EVENT_SRC_CH0G	channel 0 capture or compare event generation,TIMERx(x=0..4,7..13)
TIMER_EVENT_SRC_CH1G	channel 1 capture or compare event generation,TIMERx(x=0..4,7,8,11)
TIMER_EVENT_SRC_CH2G	channel 2 capture or compare event generation,TIMERx(x=0..4,7)
TIMER_EVENT_SRC_C	channel 3 capture or compare event generation,TIMERx(x=0..4,7)

H3G	
TIMER_EVENT_SRC_CMTG	channel commutation event generation,TIMERx(x=0,7)
TIMER_EVENT_SRC_TRGG	trigger event generation,TIMERx(x=0..4,7,8,11)
TIMER_EVENT_SRC_BRKG	break event generation,TIMERx(x=0,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* software generate update event*/
timer_event_software_generate (TIMER0, TIMER_EVENT_SRC_UPG);
```

timer_break_struct_para_init

The description of timer_break_struct_para_init is shown as below:

Table 3-466. Function timer_break_struct_para_init

Function name	timer_break_struct_para_init
Function prototype	void timer_break_struct_para_init(timer_break_parameter_struct* breakpara);
Function descriptions	initialize the parameters of TIMER break parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to ***Table 3-438. Structure timer_break_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER break parameter struct with a default value */
timer_break_parameter_struct timer_breakpara;
timer_break_struct_para_init(timer_breakpara);
```

timer_break_config

The description of timer_break_config is shown as below:

Table 3-467. Function timer_break_config

Function name	timer_break_config
Function prototype	void timer_break_config(uint32_t timer_periph, timer_break_parameter_struct* breakpara);
Function descriptions	configure TIMER break function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to ***Table 3-438. Structure timer_break_parameter_struct***.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 break function */
timer_break_parameter_struct timer_breakpara;
timer_breakpara.runoffstate          = TIMER_ROS_STATE_DISABLE;
timer_breakpara.ideloffstate         = TIMER_IOS_STATE_DISABLE ;
timer_breakpara.deadtime             = 255;
timer_breakpara.breakpolarity        = TIMER_BREAK_POLARITY_LOW;
timer_breakpara.outputautostate      = TIMER_OUTAUTO_ENABLE;
timer_breakpara.protectmode          = TIMER_CCHP_PROT_0;
timer_breakpara.breakstate           = TIMER_BREAK_ENABLE;
timer_break_config(TIMER0, &timer_breakpara);
```

timer_break_enable

The description of timer_break_enable is shown as below:

Table 3-468. Function timer_break_enable

Function name	timer_break_enable

Function prototype	void timer_break_enable(uint32_t timer_periph);
Function descriptions	enable TIMER break function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable TIMER0 break function*/

timer_break_enable (TIMER0);

timer_break_disable

The description of timer_break_disable is shown as below:

Table 3-469. Function timer_break_disable

Function name	timer_break_disable
Function prototype	void timer_break_disable(uint32_t timer_periph);
Function descriptions	disable TIMER break function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TIMER0 break function*/

timer_break_ disable (TIMER0);

timer_automatic_output_enable

The description of timer_automatic_output_enable is shown as below:

Table 3-470. Function timer_automatic_output_enable

Function name	timer_automatic_output_enable
Function prototype	void timer_automatic_output_enable(uint32_t timer_periph);
Function descriptions	enable TIMER output automatic function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 output automatic function */
timer_automatic_output_enable (TIMER0);
```

timer_automatic_output_disable

The description of timer_automatic_output_disable is shown as below:

Table 3-471. Function timer_automatic_output_disable

Function name	timer_automatic_output_ disable
Function prototype	void timer_automatic_output_ disable (uint32_t timer_periph);
Function descriptions	disable TIMER output automatic function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 output automatic function */
timer_automatic_output_disable (TIMER0);
```

timer_primary_output_config

The description of timer_primary_output_config is shown as below:

Table 3-472. Function timer_primary_output_config

Function name	timer_primary_output_config
Function prototype	void timer_primary_output_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	configure TIMER primary output function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable TIMER0 primary output function */

timer_primary_output_config (TIMER0, ENABLE);

timer_channel_control_shadow_config

The description of timer_channel_control_shadow_config is shown as below:

Table 3-473. Function timer_channel_control_shadow_config

Function name	timer_channel_control_shadow_config
Function prototype	void timer_channel_control_shadow_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	channel commutation control shadow register enable
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function

DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* channel capture/compare control shadow register enable */

timer_channel_control_shadow_config (TIMER0, ENABLE);

timer_channel_control_shadow_update_config

The description of timer_channel_control_shadow_update_config is shown as below:

Table 3-474. Function timer_channel_control_shadow_update_config

Function name	timer_channel_control_shadow_update_config
Function prototype	void timer_channel_control_shadow_update_config(uint32_t timer_periph, uint8_t ccuctl);
Function descriptions	configure commutation control shadow register update control
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
ccuctl	channel control shadow register update control
TIMER_UPDATECTL_CCU	the shadow registers update by when CMTG bit is set
TIMER_UPDATECTL_CCUTRI	the shadow registers update by when CMTG bit is set or an rising edge of TRGI occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel control shadow register update when CMTG bit is set */

timer_channel_control_shadow_update_config (TIMER0, TIMER_UPDATECTL_CCU);

timer_channel_output_struct_para_init

The description of timer_channel_output_struct_para_init is shown as below:

Table 3-475. Function timer_channel_output_struct_para_init

Function name	timer_channel_output_struct_para_init
Function prototype	void timer_channel_output_struct_para_init(timer_oc_parameter_struct* ocpara);
Function descriptions	initialize the parameters of TIMER channel output parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
ocpara	TIMER channel output parameter struct, the structure members can refer to *Table 3-439. Structure timer_oc_parameter_struct*.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER channel output parameter struct with a default value */
timer_oc_parameter_struct timer_ocinitpara;
timer_channel_output_struct_para_init(timer_ocinitpara);
```

timer_channel_output_config

The description of timer_channel_output_config is shown as below:

Table 3-476. Function timer_channel_output_config

Function name	timer_channel_output_config
Function prototype	void timer_channel_output_config(uint32_t timer_periph, uint16_t channel, timer_oc_parameter_struct* ocpara);
Function descriptions	configure TIMER channel output function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	

ocpara	TIMER channel output parameter struct, the structure members can refer to *Table 3-439. Structure timer_oc_parameter_struct*.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output function */
timer_oc_parameter_struct timer_ocinitpara;
timer_ocinitpara.outputstate   = TIMER_CCX_ENABLE;
timer_ocinitpara.outputnstate = TIMER_CCXN_ENABLE;
timer_ocinitpara.ocpolarity     = TIMER_OC_POLARITY_HIGH;
timer_ocinitpara.ocnpolarity    = TIMER_OCN_POLARITY_HIGH;
timer_ocinitpara.ocidlestate    = TIMER_OC_IDLE_STATE_HIGH;
timer_ocinitpara.ocnidlestate = TIMER_OCN_IDLE_STATE_LOW;
timer_channel_output_config(TIMER0, TIMER_CH_0, &timer_ocinitpara);
```

timer_channel_output_mode_config

The description of timer_channel_output_mode_config is shown as below:

Table 3-477. Function timer_channel_output_mode_config

Function name	timer_channel_output_mode_config
Function prototype	void timer_channel_output_mode_config(uint32_t timer_periph, uint16_t channel, uint16_t ocmode);
Function descriptions	configure TIMER channel output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	
ocmode	channel output compare mode

TIMER_OC_MODE_TIMING	timing mode
TIMER_OC_MODE_ACTIVE	set the channel output
TIMER_OC_MODE_INACTIVE	clear the channel output
TIMER_OC_MODE_TOGGLE	toggle on match
TIMER_OC_MODE_LOW	force low mode
TIMER_OC_MODE_HIGH	force high mode
TIMER_OC_MODE_PWM0	PWM mode 0
TIMER_OC_MODE_PWM1	PWM mode 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel PWM 0 mode */
timer_channel_output_mode_config(TIMER0, TIMER_CH_0, TIMER_OC_MODE_PWM0);
```

timer_channel_output_pulse_value_config

The description of timer_channel_output_pulse_value_config is shown as below:

Table 3-478. Function timer_channel_output_pulse_value_config

Function name	timer_channel_output_pulse_value_config
Function prototype	void timer_channel_output_pulse_value_config(uint32_t timer_periph, uint16_t channel, uint32_t pulse);
Function descriptions	configure TIMER channel output pulse value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))

TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	
pulse	channel output pulse value(0~65535)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 output pulse value */

timer_channel_output_pulse_value_config(TIMER0, TIMER_CH_0, 399);

timer_channel_output_shadow_config

The description of timer_channel_output_shadow_config is shown as below:

Table 3-479. Function timer_channel_output_shadow_config

Function name	timer_channel_output_shadow_config
Function prototype	void timer_channel_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t ocshadow);
Function descriptions	configure TIMER channel output shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	
ocshadow	channel output shadow state
TIMER_OC_SHADOW_ENABLE	channel output shadow state enable
TIMER_OC_SHADOW_DISABLE	channel output shadow state disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/*configure TIMER0 channel 0 output shadow function */

timer_channel_output_shadow_config (TIMER0, TIMER_CH_0,
TIMER_OC_SHADOW_ENABLE);

timer_channel_output_fast_config

The description of timer_channel_output_fast_config is shown as below:

Table 3-480. Function timer_channel_output_fast_config

Function name	timer_channel_output_fast_config
Function prototype	void timer_channel_output_fast_config(uint32_t timer_periph, uint16_t channel, uint16_t ocfast);
Function descriptions	configure TIMER channel output fast function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	
ocfast	channel output fast function
TIMER_OC_FAST_ENABLE	channel output fast function enable
TIMER_OC_FAST_DISABLE	channel output fast function disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 output fast function */

timer_channel_output_fast_config (TIMER0, TIMER_CH_0, TIMER_OC_FAST_ENABLE);

timer_channel_output_clear_config

The description of timer_channel_output_clear_config is shown as below:

Table 3-481. Function timer_channel_output_clear_config

Function name	timer_channel_output_clear_config
Function prototype	void timer_channel_output_clear_config(uint32_t timer_periph, uint16_t channel, uint16_t occlear);
Function descriptions	configure TIMER channel output clear function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER periphera
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	
occlear	channel output clear function
TIMER_OC_CLEAR_ENABLE	channel output clear function enable
TIMER_OC_CLEAR_DISABLE	channel output clear function disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output clear function */

timer_channel_output_clear_config (TIMER0, TIMER_CH_0,
TIMER_OC_CLEAR_ENABLE);
```

timer_channel_output_polarity_config

The description of timer_channel_output_polarity_config is shown as below:

Table 3-482. Function timer_channel_output_polarity_config

Function name	timer_channel_output_polarity_config
Function prototype	void timer_channel_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocpolarity);
Function descriptions	configure TIMER channel output polarity
Precondition	-
The called functions	-

Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	
ocpolarity	channel output polarity
TIMER_OC_POLARITY_HIGH	channel output polarity is high
TIMER_OC_POLARITY_LOW	channel output polarity is low
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 output polarity */

timer_channel_output_polarity_config (TIMER0, TIMER_CH_0,
TIMER_OC_POLARITY_HIGH);

timer_channel_complementary_output_polarity_config

The description of timer_channel_complementary_output_polarity_config is shown as below:

Table 3-483. Function timer_channel_complementary_output_polarity_config

Function name	timer_channel_complementary_output_polarity_config
Function prototype	void timer_channel_complementary_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnpolarity);
Function descriptions	configure TIMER channel complementary output polarity
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0
TIMER_CH_1	TIMER channel 1

TIMER_CH_2	TIMER channel 2
Input parameter{in}	
ocpolarity	channel complementary output polarity
TIMER_OCN_POLARITY_HIGH	channel complementary output polarity is high
TIMER_OCN_POLARITY_LOW	channel complementary output polarity is low
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 complementary output polarity */

timer_channel_complementary_output_polarity_config (TIMER0, TIMER_CH_0, TIMER_OCN_POLARITY_HIGH);

timer_channel_output_state_config

The description of timer_channel_output_state_config is shown as below:

Table 3-484. Function timer_channel_output_state_config

Function name	timer_channel_output_state_config
Function prototype	void timer_channel_output_state_config(uint32_t timer_periph, uint16_t channel, uint32_t state);
Function descriptions	configure TIMER channel enable state
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	
state	TIMER channel enable state
TIMER_CCX_ENABLE	channel enable
TIMER_CCX_DISABLE	channel disable
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* configure TIMER0 channel 0 enable state */

timer_channel_output_state_config (TIMER0, TIMER_CH_0, TIMER_CCX_ENABLE);

timer_channel_complementary_output_state_config

The description of timer_channel_complementary_output_state_config is shown as below:

Table 3-485. Function timer_channel_complementary_output_state_config

Function name	timer_channel_complementary_output_state_config
Function prototype	void timer_channel_complementary_output_state_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnstate);
Function descriptions	configure TIMER channel complementary output enable state
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0
TIMER_CH_1	TIMER channel 1
TIMER_CH_2	TIMER channel 2
Input parameter{in}	
state	TIMER channel complementary output enable state
TIMER_CCXN_ENABLE	channel complementary enable
TIMER_CCXN_DISABLE	channel complementary disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 complementary output enable state */

timer_channel_complementary_output_state_config (TIMER0, TIMER_CH_0, TIMER_CCXN_ENABLE);

timer_channel_input_struct_para_init

The description of timer_channel_input_struct_para_init is shown as below:

Table 3-486. Function timer_channel_input_struct_para_init

Function name	timer_channel_input_struct_para_init
Function prototype	void timer_channel_input_struct_para_init(timer_ic_parameter_struct* icpara);
Function descriptions	initialize the parameters of TIMER channel input parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
icpara	TIMER channel intput parameter struct, the structure members can refer to *Table 3-440. Structure timer_ic_parameter_struct*.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER channel input parameter struct with a default value */
timer_ic_parameter_struct timer_icinitpara;
timer_channel_input_struct_para_init(timer_icinitpara);
```

timer_input_capture_config

The description of timer_input_capture_config is shown as below:

Table 3-487. Function timer_input_capture_config

Function name	timer_input_capture_config
Function prototype	void timer_input_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpara);
Function descriptions	configure TIMER input capture parameter
Precondition	-
The called functions	timer_channel_input_capture_prescaler_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))

TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	
icpara	TIMER channel intput parameter struct, the structure members can refer to *Table 3-440. Structure timer_ic_parameter_struct*.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 input capture parameter    */
timer_ic_parameter_struct timer_icinitpara;
timer_icinitpara.icpolarity    = TIMER_IC_POLARITY_RISING;
timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTI;
timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;
timer_icinitpara.icfilter        = 0x0;
timer_input_capture_config(TIMER0, TIMER_CH_0, &timer_icinitpara);
```

timer_channel_input_capture_prescaler_config

The description of timer_channel_input_capture_prescaler_config is shown as below:

Table 3-488. Function timer_channel_input_capture_prescaler_config

Function name	timer_channel_input_capture_prescaler_config
Function prototype	void timer_channel_input_capture_prescaler_config(uint32_t timer_periph, uint16_t channel, uint16_t prescaler);
Function descriptions	configure TIMER channel input capture prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Input parameter{in}	

prescaler	channel input capture prescaler value
TIMER_IC_PSC_DIV1	no prescaler
TIMER_IC_PSC_DIV2	divided by 2
TIMER_IC_PSC_DIV4	divided by 4
TIMER_IC_PSC_DIV8	divided by 8
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 channel 0 input capture prescaler value */

timer_channel_input_capture_prescaler_config (TIMER0, TIMER_CH_0, TIMER_IC_PSC_DIV2);

timer_channel_capture_value_register_read

The description of timer_channel_capture_value_register_read is shown as below:

Table 3-489. Function timer_channel_capture_value_register_read

Function name	timer_channel_capture_value_register_read
Function prototype	uint32_t timer_channel_capture_value_register_read(uint32_t timer_periph, uint16_t channel);
Function descriptions	read TIMER channel capture compare register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0..4,7..13))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4,7,8,11))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4,7))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0..4,7))
Output parameter{out}	
-	-
Return value	
uint32_t	channel capture compare register value (0x0000~0xFFFF)

Example:

/* read TIMER0 channel 0 capture compare register value */

uint32_t ch0_value = 0;

ch0_value = timer_channel_capture_value_register_read (TIMER0, TIMER_CH_0);

timer_input_pwm_capture_config

The description of timer_input_pwm_capture_config is shown as below:

Table 3-490. Function timer_input_pwm_capture_config

Function name	timer_input_pwm_capture_config
Function prototype	void timer_input_pwm_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpwm);
Function descriptions	configure TIMER input pwm capture function
Precondition	-
The called functions	timer_channel_input_capture_prescaler_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
TIMER_CH_0	TIMER channel 0
TIMER_CH_1	TIMER channel 1
Input parameter{in}	
icpwm	TIMER channel intput pwm parameter struct, the structure members can refer to *Table 3-440. Structure timer_ic_parameter_struct*.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 input pwm capture parameter */
timer_ic_parameter_struct timer_icinitpara;
timer_icinitpara.icpolarity = TIMER_IC_POLARITY_RISING;
timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTI;
timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;
timer_icinitpara.icfilter = 0x0;
timer_input_pwm_capture_config (TIMER0, TIMER_CH_0, &timer_icinitpara);
```

timer_hall_mode_config

The description of timer_hall_mode_config is shown as below:

Table 3-491. Function timer_hall_mode_config

Function name	timer_hall_mode_config
Function prototype	void timer_hall_mode_config(uint32_t timer_periph, uint8_t hallmode);
Function descriptions	configure TIMER hall sensor mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7)	TIMER peripheral selection
Input parameter{in}	
hallmode	TIMER hall sensor mode state
TIMER_HALLINTERFACE_ENABLE	TIMER hall sensor mode enable
TIMER_HALLINTERFACE_DISABLE	TIMER hall sensor mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 hall sensor mode */

timer_hall_mode_config (TIMER0, TIMER_HALLINTERFACE_ENABLE);

timer_input_trigger_source_select

The description of timer_input_trigger_source_select is shown as below:

Table 3-492. Function timer_input_trigger_source_select

Function name	timer_input_trigger_source_select
Function prototype	void timer_input_trigger_source_select(uint32_t timer_periph, uint32_t intrigger);
Function descriptions	select TIMER input trigger source
Precondition	SMC[2:0] = 000
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
intrigger	trigger selection
TIMER_SMCFG_TRGSEL_ITI0	Internal trigger input 0 (ITI0, TIMERx(x=0..4,7,8,11))
TIMER_SMCFG_TRGS	Internal trigger input 0 (ITI1, TIMERx(x=0..4,7,8,11))

EL_ITI1	
TIMER_SMCFG_TRGSEL_ITI2	Internal trigger input 0 (ITI2, TIMERx(x=0..4,7,8,11))
TIMER_SMCFG_TRGSEL_ITI3	Internal trigger input 0 (ITI3, TIMERx(x=0..4,7,8,11))
TIMER_SMCFG_TRGSEL_CI0F_ED	CI0 edge flag (CI0F_ED, TIMERx(x=0..4,7,8,11))
TIMER_SMCFG_TRGSEL_CI0FE0	channel 0 input Filtered output (CI0FE0, TIMERx(x=0..4,7,8,11))
TIMER_SMCFG_TRGSEL_CI1FE1	channel 1 input Filtered output (CI1FE1, TIMERx(x=0..4,7,8,11))
TIMER_SMCFG_TRGSEL_ETIFP	External trigger input filter output(ETIFP, TIMERx(x=0..4,7))
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* select TIMER0 input trigger source */

timer_input_trigger_source_select (TIMER0, TIMER_SMCFG_TRGSEL_ITI0);

timer_master_output_trigger_source_select

The description of timer_master_output_trigger_source_select is shown as below:

Table 3-493. Function timer_master_output_trigger_source_select

Function name	timer_master_output_trigger_source_select
Function prototype	void timer_master_output_trigger_source_select(uint32_t timer_periph, uint32_t outrigger);
Function descriptions	select TIMER master mode output trigger source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
outrigger	master mode control
TIMER_TRI_OUT_SRC_RESET	Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset
TIMER_TRI_OUT_SRC	Enable. This mode is useful to start several timers at the same time or to

_ENABLE	control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected.
TIMER_TRI_OUT_SRC_UPDATE	Update. In this mode the master mode controller selects the update event as TRGO.
TIMER_TRI_OUT_SRC_CC0	Capture/compare pulse. In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred in channel 0.
TIMER_TRI_OUT_SRC_O0CPRE	Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO.
TIMER_TRI_OUT_SRC_O1CPRE	Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO.
TIMER_TRI_OUT_SRC_O2CPRE	Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO.
TIMER_TRI_OUT_SRC_O3CPRE	Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER0 master mode output trigger source */
timer_master_output_trigger_source_select (TIMER0, TIMER_TRI_OUT_SRC_RESET);
```

timer_slave_mode_select

The description of timer_slave_mode_select is shown as below:

Table 3-494. Function timer_slave_mode_select

Function name	timer_slave_mode_select
Function prototype	void timer_slave_mode_select(uint32_t timer_periph, uint32_t slavemode);
Function descriptions	select TIMER slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	
slavemode	slave mode
TIMER_SLAVE_MODE_DISABLE	slave mode disable

TIMER_QUAD_DECODER_MODE0	quadrature decoder mode 0
TIMER_QUAD_DECODER_MODE1	quadrature decoder mode 1
TIMER_QUAD_DECODER_MODE2	quadrature decoder mode 2
TIMER_SLAVE_MODE_RESTART	restart mode
TIMER_SLAVE_MODE_PAUSE	pause mode
TIMER_SLAVE_MODE_EVENT	event mode
TIMER_SLAVE_MODE_EXTERNAL0	external clock mode 0
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER0 slave mode */
timer_slave_mode_select (TIMER0, TIMER_QUAD_DECODER_MODE0);
```

timer_master_slave_mode_config

The description of timer_master_slave_mode_config is shown as below:

Table 3-495. Function timer_master_slave_mode_config

Function name	timer_master_slave_mode_config
Function prototype	void timer_master_slave_mode_config(uint32_t timer_periph, uint8_t masterslave);
Function descriptions	configure TIMER master slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	
masterslave	master slave mode state
TIMER_MASTER_SLAVE_MODE_ENABLE	master slave mode enable
TIMER_MASTER_SLAVE_MODE_DISABLE	master slave mode disable

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 master slave mode  */
timer_master_slave_mode_config (TIMER0, TIMER_MASTER_SLAVE_MODE_ENABLE);
```

timer_external_trigger_config

The description of timer_external_trigger_config is shown as below:

Table 3-496. Function timer_external_trigger_config

Function name	timer_external_trigger_config
Function prototype	void timer_external_trigger_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER external trigger input
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7)	TIMER peripheral selection
Input parameter{in}	
extprescaler	external trigger prescaler
TIMER_EXT_TRI_PSC_OFF	no divided
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
Input parameter{in}	
expolarity	external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active
TIMER_ETP_RISING	active high or rising edge active
Input parameter{in}	
extfilter	external trigger filter control(0~15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 external trigger input */

timer_external_trigger_config (TIMER0, TIMER_EXT_TRI_PSC_DIV2, TIMER_ETP_FALLING, 10);

timer_quadrature_decoder_mode_config

The description of timer_quadrature_decoder_mode_config is shown as below:

Table 3-497. Function timer_quadrature_decoder_mode_config

Function name	timer_quadrature_decoder_mode_config
Function prototype	void timer_quadrature_decoder_mode_config(uint32_t timer_periph, uint32_t decomode, uint16_t ic0polarity, uint16_t ic1polarity);
Function descriptions	configure TIMER quadrature decoder mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7)	TIMER peripheral selection
Input parameter{in}	
decomode	quadrature decoder mode
TIMER_QUAD_DECODER_MODE0	counter counts on CI0FE0 edge depending on CI1FE1 level
TIMER_QUAD_DECODER_MODE1	counter counts on CI1FE1 edge depending on CI0FE0 level
TIMER_QUAD_DECODER_MODE2	counter counts on both CI0FE0 and CI1FE1 edges depending on the level of the other input
Input parameter{in}	
ic0polarity	IC0 polarity
TIMER_IC_POLARITY_RISING	capture rising edge
TIMER_IC_POLARITY_FALLING	capture falling edge
Input parameter{in}	
ic1polarity	IC1 polarity
TIMER_IC_POLARITY_RISING	capture rising edge
TIMER_IC_POLARITY_FALLING	capture falling edge
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* configure TIMER0 quadrature decoder mode */

timer_quadrature_decoder_mode_config (TIMER0, TIMER_QUAD_DECODER_MODE0, TIMER_IC_POLARITY_RISING, TIMER_IC_POLARITY_RISING);

timer_internal_clock_config

The description of timer_internal_clock_config is shown as below:

Table 3-498. Function timer_internal_clock_config

Function name	timer_internal_clock_config
Function prototype	void timer_internal_clock_config(uint32_t timer_periph);
Function descriptions	configure TIMER internal clock mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 internal clock mode */

timer_internal_clock_config (TIMER0);

timer_internal_trigger_as_external_clock_config

The description of timer_internal_trigger_as_external_clock_config is shown as below:

Table 3-499. Function timer_internal_trigger_as_external_clock_config

Function name	timer_internal_trigger_as_external_clock_config
Function prototype	void timer_internal_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t intrigger);
Function descriptions	configure TIMER the internal trigger as external clock input
Precondition	-
The called functions	timer_input_trigger_source_select
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection

Input parameter{in}	
intrigger	trigger selection
TIMER_SMCFG_TRGSEL_ITI0	Internal trigger input 0 (ITI0)
TIMER_SMCFG_TRGSEL_ITI1	Internal trigger input 0 (ITI1)
TIMER_SMCFG_TRGSEL_ITI2	Internal trigger input 0 (ITI2)
TIMER_SMCFG_TRGSEL_ITI3	Internal trigger input 0 (ITI3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 the internal trigger ITI0 as external clock input */
timer_internal_trigger_as_external_clock_config (TIMER0, TIMER_SMCFG_TRGSEL_ITI0);
```

timer_external_trigger_as_external_clock_config

The description of timer_external_trigger_as_external_clock_config is shown as below:

Table 3-500. Function timer_external_trigger_as_external_clock_config

Function name	timer_external_trigger_as_external_clock_config
Function prototype	void timer_external_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t extrigger, uint16_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external trigger as external clock input
Precondition	-
The called functions	timer_input_trigger_source_select
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	
extrigger	external trigger selection
TIMER_SMCFG_TRGSEL_CI0F_ED	CI0 edge flag (CI0F_ED)
TIMER_SMCFG_TRGSEL_CI0FE0	channel 0 input Filtered output (CI0FE0)
TIMER_SMCFG_TRGSEL_CI1FE1	channel 1 input Filtered output (CI1FE1)
Input parameter{in}	
expolarity	external trigger polarity

TIMER_IC_POLARITY_RISING	active high or rising edge active
TIMER_IC_POLARITY_FALLING	active low or falling edge active
Input parameter{in}	
extfilter	external trigger filter control(0~15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 the external trigger CI0FE0 as external clock input */

timer_external_trigger_as_external_clock_config (TIMER0, TIMER_SMCFG_TRGSEL_CI0FE0, TIMER_IC_POLARITY_RISING, 0);

timer_external_clock_mode0_config

The description of timer_external_clock_mode0_config is shown as below:

Table 3-501. Function timer_external_clock_mode0_config

Function name	timer_external_clock_mode0_config
Function prototype	void timer_external_clock_mode0_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external clock mode0
Precondition	-
The called functions	timer_external_trigger_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7,8,11)	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI external trigger prescaler
TIMER_EXT_TRI_PSC_OFF	no divided
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
Input parameter{in}	
expolarity	ETI external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active

TIMER_ETP_RISING	active high or rising edge active
Input parameter{in}	
extfilter	ETI external trigger filter control(0~15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 the external clock mode0 */

timer_external_clock_mode0_config (TIMER0, TIMER_EXT_TRI_PSC_DIV2, TIMER_ETP_FALLING, 0);

timer_external_clock_mode1_config

The description of timer_external_clock_mode1_config is shown as below:

Table 3-502. Function timer_external_clock_mode1_config

Function name	timer_external_clock_mode1_config
Function prototype	void timer_external_clock_mode1_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external clock mode1
Precondition	-
The called functions	timer_external_trigger_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7)	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI external trigger prescaler
TIMER_EXT_TRI_PSC_OFF	no divided
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
Input parameter{in}	
expolarity	ETI external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active
TIMER_ETP_RISING	active high or rising edge active
Input parameter{in}	
extfilter	ETI external trigger filter control(0~15)

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER0 the external clock mode1 */

timer_external_clock_mode1_config (TIMER0, TIMER_EXT_TRI_PSC_DIV2, TIMER_ETP_FALLING, 0);

timer_external_clock_mode1_disable

The description of timer_external_clock_mode1_disable is shown as below:

Table 3-503. Function timer_external_clock_mode1_disable

Function name	timer_external_clock_mode1_disable
Function prototype	void timer_external_clock_mode1_disable(uint32_t timer_periph);
Function descriptions	disable TIMER the external clock mode1
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable TIMER0 the external clock mode1 */

timer_external_clock_mode1_disable (TIMER0);

timer_write_chxval_register_config

The description of timer_write_chxval_register_config is shown as below:

Table 3-504. Function timer_write_chxval_register_config

Function name	timer_write_chxval_register_config
Function prototype	void timer_write_chxval_register_config(uint32_t timer_periph, uint16_t ccsel);
Function descriptions	configure TIMER write CHxVAL register selection
Precondition	-
The called functions	-

Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4,7..13)	TIMER peripheral selection
Input parameter{in}	
ccsel	write CHxVAL register selection
TIMER_CHVSEL_DISABLE	no effect
TIMER_CHVSEL_ENABLE	when write the CHxVAL register, if the write value is same as the CHxVAL value, the write access is ignored
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 write CHxVAL register selection */
timer_write_chxval_register_config(TIMER0, TIMER_CHVSEL_ENABLE);
```

timer_output_value_selection_config

The description of timer_output_value_selection_config is shown as below:

Table 3-505. Function timer_output_value_selection_config

Function name	timer_output_value_selection_config
Function prototype	void timer_output_value_selection_config(uint32_t timer_periph, uint16_t outsel);
Function descriptions	configure TIMER output value selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx (x=0,7)	TIMER peripheral selection
Input parameter{in}	
outsel	output value selection
TIMER_OUTSEL_DISABLE	no effect
TIMER_OUTSEL_ENABLE	if POEN and IOS is 0, the output disabled
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure TIMER output value selection */

timer_output_value_selection_config(TIMER0, TIMER_OUTSEL_ENABLE);

timer_interrupt_enable

The description of timer_interrupt_enable is shown as below:

Table 3-506. Function timer_interrupt_enable

Function name	timer_interrupt_enable
Function prototype	void timer_interrupt_enable(uint32_t timer_periph, uint32_t interrupt);
Function descriptions	enable the TIMER interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
interrupt	timer interrupt enable source
TIMER_INT_UP	update interrupt enable, TIMERx (x=0..13)
TIMER_INT_CH0	channel 0 interrupt enable, TIMERx (x=0..4,7..13)
TIMER_INT_CH1	channel 1 interrupt enable, TIMERx (x=0..4,7,8,11)
TIMER_INT_CH2	channel 2 interrupt enable, TIMERx (x=0..4,7)
TIMER_INT_CH3	channel 3 interrupt enable , TIMERx (x=0..4,7)
TIMER_INT_CMT	commutation interrupt enable, TIMERx (x=0,7)
TIMER_INT_TRG	trigger interrupt enable, TIMERx (x=0..4,7,8,11)
TIMER_INT_BRK	break interrupt enable, TIMERx (x=0,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the TIMER0 update interrupt */

timer_interrupt_enable (TIMER0, TIMER_INT_UP);

timer_interrupt_disable

The description of timer_interrupt_disable is shown as below:

Table 3-507. Function timer_interrupt_disable

Function name	timer_interrupt_ disable
Function prototype	void timer_interrupt_ disable (uint32_t timer_periph, uint32_t interrupt);
Function descriptions	disable the TIMER interrupt

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
interrupt	timer interrupt disable source
TIMER_INT_UP	update interrupt disable, TIMERx(x=0..13)
TIMER_INT_CH0	channel 0 interrupt disable, TIMERx(x=0..4,7..13)
TIMER_INT_CH1	channel 1 interrupt disable, TIMERx(x=0..4,7,8,11)
TIMER_INT_CH2	channel 2 interrupt disable, TIMERx(x=0..4,7)
TIMER_INT_CH3	channel 3 interrupt disable, TIMERx(x=0..4,7)
TIMER_INT_CMT	commutation interrupt disable, TIMERx(x=0,7)
TIMER_INT_TRG	trigger interrupt disable, TIMERx(x=0..4,7,8,11)
TIMER_INT_BRK	break interrupt disable, TIMERx(x=0,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMER0 update interrupt */
timer_interrupt_disable (TIMER0, TIMER_INT_UP);
```

timer_interrupt_flag_get

The description of timer_interrupt_flag_get is shown as below:

Table 3-508. Function timer_interrupt_flag_get

Function name	timer_interrupt_flag_get
Function prototype	FlagStatus timer_interrupt_flag_get(uint32_t timer_periph, uint32_t interrupt);
Function descriptions	get timer interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
interrupt	the timer interrupt bits
TIMER_INT_FLAG_UP	update interrupt flag,TIMERx(x=0..13)
TIMER_INT_FLAG_CH0	channel 0 interrupt flag,TIMERx(x=0..4,7..13)
TIMER_INT_FLAG_CH1	channel 1 interrupt flag,TIMERx(x=0..4,7,8,11)

TIMER_INT_FLAG_CH2	channel 2 interrupt flag,TIMERx(x=0..4,7)
TIMER_INT_FLAG_CH3	channel 3 interrupt flag,TIMERx(x=0..4,7)
TIMER_INT_FLAG_CMT	channel commutation interrupt flag,TIMERx(x=0,7)
TIMER_INT_FLAG_TRG	trigger interrupt flag,TIMERx(x=0,7,8,11)
TIMER_INT_FLAG_BRK	break interrupt flag,TIMERx(x=0,7)
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

/* get TIMER0 update interrupt flag */

FlagStatus Flag_ interrupt = RESET;

Flag_ interrupt = timer_interrupt_flag_get (TIMER0, TIMER_INT_FLAG_UP);

timer_interrupt_flag_clear

The description of timer_interrupt_flag_clear is shown as below:

Table 3-509. Function timer_interrupt_flag_clear

Function name	timer_interrupt_flag_clear
Function prototype	void timer_interrupt_flag_clear(uint32_t timer_periph, uint32_t interrupt);
Function descriptions	clear TIMER interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
interrupt	the timer interrupt bits
TIMER_INT_FLAG_UP	update interrupt flag,TIMERx(x=0..13)
TIMER_INT_FLAG_CH0	channel 0 interrupt flag,TIMERx(x=0..4,7..13)
TIMER_INT_FLAG_CH1	channel 1 interrupt flag,TIMERx(x=0..4,7,8,11)
TIMER_INT_FLAG_CH2	channel 2 interrupt flag,TIMERx(x=0..4,7)
TIMER_INT_FLAG_CH3	channel 3 interrupt flag,TIMERx(x=0..4,7)
TIMER_INT_FLAG_CMT	channel commutation interrupt flag,TIMERx(x=0,7)
TIMER_INT_FLAG_TRG	trigger interrupt flag,TIMERx(x=0,7,8,11)
TIMER_INT_FLAG_BRK	break interrupt flag,TIMERx(x=0,7)
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* clear TIMER0 update interrupt flag */
timer_interrupt_flag_clear (TIMER0, TIMER_INT_FLAG_UP);
```

timer_flag_get

The description of timer_flag_get is shown as below:

Table 3-510. Function timer_flag_get

Function name	timer_flag_get
Function prototype	FlagStatus timer_flag_get(uint32_t timer_periph, uint32_t flag);
Function descriptions	get TIMER flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
TIMER_FLAG_UP	update flag,TIMERx(x=0..13)
TIMER_FLAG_CH0	channel 0 flag,TIMERx(x=0..4,7..13)
TIMER_FLAG_CH1	channel 1 flag,TIMERx(x=0..4,7,8,11)
TIMER_FLAG_CH2	channel 2 flag,TIMERx(x=0..4,7)
TIMER_FLAG_CH3	channel 3 flag,TIMERx(x=0..4,7)
TIMER_FLAG_CMT	channel commutation flag,TIMERx(x=0,7)
TIMER_FLAG_TRG	trigger flag,TIMERx(x=0,7,8,11)
TIMER_FLAG_BRK	break flag,TIMERx(x=0,7)
TIMER_FLAG_CH0O	channel 0 overcapture flag,TIMERx(x=0..4,7..11)
TIMER_FLAG_CH1O	channel 1 overcapture flag,TIMERx(x=0..4,7,8,11)
TIMER_FLAG_CH2O	channel 2 overcapture flag,TIMERx(x=0..4,7)
TIMER_FLAG_CH3O	channel 3 overcapture flag,TIMERx(x=0..4,7)
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TIMER0 update flags */
FlagStatus Flag_status = RESET;
```

Flag_status = timer_flag_get (TIMER0, TIMER_FLAG_UP);

timer_flag_clear

The description of timer_flag_clear is shown as below:

Table 3-511. Function timer_flag_clear

Function name	timer_flag_clear
Function prototype	void timer_flag_clear(uint32_t timer_periph, uint32_t flag);
Function descriptions	clear TIMER flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
TIMER_FLAG_UP	update flag,TIMERx(x=0..13)
TIMER_FLAG_CH0	channel 0 flag,TIMERx(x=0..4,7..13)
TIMER_FLAG_CH1	channel 1 flag,TIMERx(x=0..4,7,8,11)
TIMER_FLAG_CH2	channel 2 flag,TIMERx(x=0..4,7)
TIMER_FLAG_CH3	channel 3 flag,TIMERx(x=0..4,7)
TIMER_FLAG_CMT	channel commutation flag,TIMERx(x=0,7)
TIMER_FLAG_TRG	trigger flag,TIMERx(x=0,7,8,11)
TIMER_FLAG_BRK	break flag,TIMERx(x=0,7)
TIMER_FLAG_CH0O	channel 0 overcapture flag,TIMERx(x=0..4,7..11)
TIMER_FLAG_CH1O	channel 1 overcapture flag,TIMERx(x=0..4,7,8,11)
TIMER_FLAG_CH2O	channel 2 overcapture flag,TIMERx(x=0..4,7)
TIMER_FLAG_CH3O	channel 3 overcapture flag,TIMERx(x=0..4,7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* clear TIMER0 update flags */

timer_flag_clear (TIMER0, TIMER_FLAG_UP);

3.21. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter ***3.21.1***, the USART

firmware functions are introduced in chapter *3.21.2*.

3.21.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-512. USART Registers

Registers	Descriptions
USART_STAT0	Status register 0
USART_DATA	Data register
USART_BAUD	Baud rate register
USART_CTL0	Control register 0
USART_CTL1	Control register 1
USART_CTL2	Control register 2
USART_GP	Guard time and prescaler register
USART_CTL3	Control register 3
USART_RT	Receiver timeout register
USART_STAT1	Status register 1
USART_CHC	Coherence control register

3.21.2. Descriptions of Peripheral functions

USART firmware functions are listed in the table shown as below:

Table 3-513. USART firmware function

Function name	Function description
usart_deinit	reset USART/UART
usart_baudrate_set	configure USART baud rate value
usart_parity_config	configure USART parity
usart_word_length_set	configure USART word length
usart_stop_bit_set	configure USART stop bit length
usart_enable	enable USART
usart_disable	disable USART
usart_transmit_config	configure USART transmitter
usart_receive_config	configure USART receiver
usart_data_first_config	data is transmitted/received with the LSB/MSB first
usart_invert_config	configure USART inverted
usart_receiver_timeout_enable	enable receiver timeout
usart_receiver_timeout_disable	disable receiver timeout
usart_receiver_timeout_threshold_config	configure receiver timeout threshold
usart_data_transmit	USART transmit data function
usart_data_receive	USART receive data function

Function name	Function description
usart_address_config	configure the address of the USART in wake up by address match mode
usart_mute_mode_enable	enable mute mode
usart_mute_mode_disable	disable mute mode
usart_mute_mode_wakeup_config	configure wakeup method in mute mode
usart_lin_mode_enable	enable LIN mode
usart_lin_mode_disable	disable LIN mode
usart_lin_break_dection_length_config	configure LIN break frame length
usart_send_break	send break frame
usart_halfduplex_enable	enable half duplex mode
usart_halfduplex_disable	disable half duplex mode
usart_synchronous_clock_enable	enable CK pin in synchronous mode
usart_synchronous_clock_disable	disable CK pin in synchronous mode
usart_synchronous_clock_config	configure USART synchronous mode parameters
usart_guard_time_config	configure guard time value in smartcard mode
usart_smartcard_mode_enable	enable smartcard mode
usart_smartcard_mode_disable	disable smartcard mode
usart_smartcard_mode_nack_enable	enable NACK in smartcard mode
usart_smartcard_mode_nack_disable	disable NACK in smartcard mode
usart_smartcard_autoretry_config	configure smartcard auto-retry number
usart_block_length_config	configure block length
usart_irda_mode_enable	enable IrDA mode
usart_irda_mode_disable	disable IrDA mode
usart_prescaler_config	configure the peripheral clock prescaler in USART IrDA low-power mode
usart_irda_lowpower_config	configure IrDA low-power
usart_hardware_flow_rts_config	configure hardware flow control RTS
usart_hardware_flow_cts_config	configure hardware flow control CTS
usart_dma_receive_config	configure USART DMA reception
usart_dma_transmit_config	configure USART DMA transmission
usart_hardware_flow_coherence_config	configure hardware flow control coherence mode
usart_flag_get	get flag in STAT0/STAT1 register
usart_flag_clear	clear flag in STAT0/STAT1 register
usart_interrupt_enable	enable USART interrupt
usart_interrupt_disable	disable USART interrupt
usart_interrupt_flag_get	get USART interrupt flag status
usart_interrupt_flag_clear	clear USART interrupt flag

Enum usart_flag_enum

Table 3-514. usart_flag_enum

Member name	Function description
USART_FLAG_CTS	CTS change flag
USART_FLAG_LBD	LIN break detected flag
USART_FLAG_TBE	transmit data buffer empty
USART_FLAG_TC	transmission complete
USART_FLAG_RBNE	read data buffer not empty
USART_FLAG_IDLE	IDLE line detected flag
USART_FLAG_ORERR	overrun error flag
USART_FLAG_NERR	noise error flag
USART_FLAG_FERR	frame error flag
USART_FLAG_PERR	parity error flag
USART_FLAG_BSY	busy flag
USART_FLAG_EB	end of block flag
USART_FLAG_RT	receiver timeout flag
USART_FLAG_EPERR	early parity error flag

Enum usart_interrupt_flag_enum

Table 3-515. usart_interrupt_flag_enum

Member name	Function description
USART_INT_FLAG_PERR	parity error interrupt flag
USART_INT_FLAG_TBE	transmitter buffer empty interrupt flag
USART_INT_FLAG_TC	transmission complete interrupt flag
USART_INT_FLAG_RBNE	read data buffer not empty interrupt flag
USART_INT_FLAG_RBNE_ORERR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_IDLE	IDLE line detected interrupt flag
USART_INT_FLAG_LBD	LIN break detected interrupt flag
USART_INT_FLAG_CTS	CTS interrupt flag
USART_INT_FLAG_ERR_ORERR	overrun error interrupt flag
USART_INT_FLAG_ERR_NERR	noise error interrupt flag
USART_INT_FLAG_ERR_FERR	frame error interrupt flag
USART_INT_FLAG_EB	end of block interrupt flag
USART_INT_FLAG_RT	receive timeout interrupt flag

Enum usart_interrupt_enum

Table 3-516. usart_interrupt_enum

Member name	Function description
USART_INT_PERR	parity error interrupt
USART_INT_TBE	transmitter buffer empty interrupt
USART_INT_TC	transmission complete interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt
USART_INT_IDLE	IDLE line detected interrupt
USART_INT_LBD	LIN break detected interrupt
USART_INT_CTS	CTS interrupt
USART_INT_ERR	error interrupt
USART_INT_EB	end of block interrupt
USART_INT_RT	receive timeout interrupt

Enum usart_invert_enum

Table 3-517. usart_invert_enum

Member name	Function description
USART_DINV_ENABLE	data bit level inversion
USART_DINV_DISABLE	data bit level not inversion
USART_TXPIN_ENABLE	TX pin level inversion
USART_TXPIN_DISABLE	TX pin level not inversion
USART_RXPIN_ENABLE	RX pin level inversion
USART_RXPIN_DISABLE	RX pin level not inversion

usart_deinit

The description of usart_deinit is shown as below:

Table 3-518. Function usart_deinit

Function name	usart_deinit
Function prototype	void usart_deinit(uint32_t usart_periph);
Function descriptions	reset USART/UART
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset USART0 */

usart_deinit (USART0);
```

usart_baudrate_set

The description of usart_baudrate_set is shown as below:

Table 3-519. Function usart_baudrate_set

Function name	usart_baudrate_set
Function prototype	void usart_baudrate_set(uint32_t usart_periph, uint32_t baudval);
Function descriptions	configure USART baud rate value
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
baudval	baud rate value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 baud rate value */

usart_baudrate_set(USART0, 115200);
```

usart_parity_config

The description of usart_parity_config is shown as below:

Table 3-520. Function usart_parity_config

Function name	usart_parity_config
Function prototype	void usart_parity_config(uint32_t usart_periph, uint32_t paritycfg);
Function descriptions	configure USART parity
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4

Input parameter{in}	
paritycfg	configure USART parity
USART_PM_NONE	no parity
USART_PM_ODD	odd parity
USART_PM_EVEN	even parity
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART parity */
usart_parity_config(USART0, USART_PM_EVEN);
```

usart_word_length_set

The description of usart_word_length_set is shown as below:

Table 3-521. Function usart_word_length_set

Function name	usart_word_length_set
Function prototype	void usart_word_length_set(uint32_t usart_periph, uint32_t wlen);
Function descriptions	configure USART word length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
wlen	USART word length
USART_WL_8BIT	8 bits
USART_WL_9BIT	9 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 word length */
usart_word_length_set(USART0, USART_WL_9BIT);
```

usart_stop_bit_set

The description of usart_stop_bit_set is shown as below:

Table 3-522. Function usart_stop_bit_set

Function name	usart_stop_bit_set
Function prototype	void usart_stop_bit_set(uint32_t usart_periph, uint32_t stblen);
Function descriptions	configure USART stop bit length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
stblen	USART stop bit
USART_STB_1BIT	1 bit
USART_STB_0_5BIT	0.5 bit, not available for UARTx(x=3,4)
USART_STB_2BIT	2 bits
USART_STB_1_5BIT	1.5 bits, not available for UARTx(x=3,4)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 stop bit length */

usart_stop_bit_set(USART0, USART_STB_1_5BIT);
```

usart_enable

The description of usart_enable is shown as below:

Table 3-523. Function usart_enable

Function name	usart_enable
Function prototype	void usart_enable(uint32_t usart_periph);
Function descriptions	enable USART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	

-	-
Return value	
-	-

Example:

/* enable USART0 */

usart_enable(USART0);

usart_disable

The description of usart_disable is shown as below:

Table 3-524. Function usart_disable

Function name	usart_disable
Function prototype	void usart_disable(uint32_t usart_periph);
Function descriptions	disable USART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable USART0 */

usart_disable(USART0);

usart_transmit_config

The description of usart_transmit_config is shown as below:

Table 3-525. Function usart_transmit_config

Function name	usart_transmit_config
Function prototype	void usart_transmit_config(uint32_t usart_periph, uint32_t txconfig);
Function descriptions	configure USART transmitter
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral

USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
txconfig	enable or disable USART transmitter
USART_TRANSMIT_ENABLE	enable USART transmission
USART_TRANSMIT_DISABLE	enable USART transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 transmitter */
usart_transmit_config(USART0,USART_TRANSMIT_ENABLE);
```

usart_receive_config

The description of usart_receive_config is shown as below:

Table 3-526. Function usart_receive_config

Function name	usart_receive_config
Function prototype	void usart_receive_config(uint32_t usart_periph, uint32_t rxconfig);
Function descriptions	configure USART receiver
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
rxconfig	enable or disable USART receiver
USART_RECEIVE_ENABLE	enable USART reception
USART_RECEIVE_DISABLE	disable USART reception
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 receiver */
usart_receive_config(USART0, USART_RECEIVE_ENABLE);
```

usart_data_first_config

The description of usart_data_first_config is shown as below:

Table 3-527. Function usart_data_first_config

Function name	usart_data_first_config
Function prototype	void usart_data_first_config(uint32_t usart_periph, uint32_t msbf);
Function descriptions	data is transmitted/received with the LSB/MSB first
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
msbf	LSB first or MSB first
USART_MSBF_LSB	LSB first
USART_MSBF_MSB	MSB first
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure LSB of data first */
usart_data_first_config(USART0, USART_MSBF_LSB);
```

usart_invert_config

The description of usart_invert_config is shown as below:

Table 3-528. Function usart_invert_config

Function name	usart_invert_config
Function prototype	void usart_invert_config(uint32_t usart_periph, usart_invert_enum invertpara);
Function descriptions	configure USART inversion
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2

Input parameter{in}	
invertpara	refer to Table 3-517. usart_invert_enum
USART_DINV_ENABLE	data bit level inversion
USART_DINV_DISABLE	data bit level not inversion
USART_TXPIN_ENABLE	TX pin level inversion
USART_TXPIN_DISABLE	TX pin level not inversion
USART_RXPIN_ENABLE	RX pin level inversion
USART_RXPIN_DISABLE	RX pin level not inversion
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART inversion */
usart_invert_config(USART0, USART_DINV_ENABLE);
```

usart_receiver_timeout_enable

The description of usart_receiver_timeout_enable is shown as below:

Table 3-529. Function usart_receiver_timeout_enable

Function name	usart_receiver_timeout_enable
Function prototype	void usart_receiver_timeout_enable(uint32_t usart_periph);
Function descriptions	enable receiver timeout
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable receiver timeout of USART */
```

usart_receiver_timeout_enable(USART0);

usart_receiver_timeout_disable

The description of usart_receiver_timeout_disable is shown as below:

Table 3-530. Function usart_receiver_timeout_disable

Function name	usart_receiver_timeout_disable
Function prototype	void usart_receiver_timeout_disable(uint32_t usart_periph);
Function descriptions	disable receiver timeout
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* disable receiver timeout of USART */

usart_receiver_timeout_disable(USART0);

usart_receiver_timeout_threshold_config

The description of usart_receiver_timeout_threshold_config is shown as below:

Table 3-531. Function usart_receiver_timeout_threshold_config

Function name	usart_receiver_timeout_threshold_config
Function prototype	void usart_receiver_timeout_threshold_config(uint32_t usart_periph, uint32_t rtimeout);
Function descriptions	configure receiver timeout threshold
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
rtimeout	timeout value
0-0xFFFFFF	timeout value
Output parameter{out}	
-	-
Return value	

-	-

Example:

/* set the receiver timeout threshold of USART0 */

usart_receiver_timeout_threshold_config(USART0, 115200*3);

usart_data_transmit

The description of usart_data_transmit is shown as below:

Table 3-532. Function usart_data_transmit

Function name	usart_data_transmit
Function prototype	void usart_data_transmit(uint32_t usart_periph, uint16_t data);
Function descriptions	USART transmit data function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
data	data of transmission
0-0x1FF	data of transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 transmit data */

usart_data_transmit(USART0, 0xAA);

usart_data_receive

The description of usart_data_receiveis shown as below:

Table 3-533. Function usart_data_receive

Function name	usart_data_receive
Function prototype	uint16_t usart_data_receive(uint32_t usart_periph);
Function descriptions	USART receive data function
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
uint16_t	data of received(0-0x1FF)

Example:

```
/* USART0 receive data */
uint16_t temp;
temp = usart_data_receive(USART0);
```

usart_address_config

The description of usart_address_config is shown as below:

Table 3-534. Function usart_address_config

Function name	usart_address_config
Function prototype	void usart_address_config(uint32_t usart_periph, uint8_t addr);
Function descriptions	configure the address of the USART in wake up by address match mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
addr	address of USART/UART
0-0xFF	address of USART/UART
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address of the USART0 */
usart_address_config(USART0, 0x00);
```

usart_mute_mode_enable

The description of usart_mute_mode_enable is shown as below:

Table 3-535. Function usart_mute_mode_enable

Function name	usart_mute_mode_enable
Function prototype	void usart_mute_mode_enable(uint32_t usart_periph);
Function descriptions	enable mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 receiver in mute mode */
usart_mute_mode_enable(USART0);
```

usart_mute_mode_disable

The description of usart_mute_mode_disable is shown as below:

Table 3-536. Function usart_mute_mode_disable

Function name	usart_mute_mode_disable
Function prototype	void usart_mute_mode_disable(uint32_t usart_periph);
Function descriptions	disable mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 receiver in mute mode */
usart_mute_mode_disable(USART0);
```

usart_mute_mode_wakeup_config

The description of usart_mute_mode_wakeup_config is shown as below:

Table 3-537. Function usart_mute_mode_wakeup_config

Function name	usart_mute_mode_wakeup_config
Function prototype	void usart_mute_mode_wakeup_config(uint32_t usart_periph, uint32_t wmethod);
Function descriptions	configure wakeup method in mute mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
wmethod	two methods be used to enter or exit the mute mode
USART_WM_IDLE	idle line
USART_WM_ADDR	address mask
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 wakeup method in mute mode */
usart_mute_mode_wakeup_config(USART0, USART_WM_IDLE);
```

usart_lin_mode_enable

The description of usart_lin_mode_enable is shown as below:

Table 3-538. Function usart_lin_mode_enable

Function name	usart_lin_mode_enable
Function prototype	void usart_lin_mode_enable(uint32_t usart_periph);
Function descriptions	enable LIN mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* USART0 LIN mode enable */

usart_lin_mode_enable(USART0);

usart_lin_mode_disable

The description of usart_lin_mode_disable is shown as below:

Table 3-539. Function usart_lin_mode_disable

Function name	usart_lin_mode_disable
Function prototype	void usart_lin_mode_disable(uint32_t usart_periph);
Function descriptions	disable LIN mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 LIN mode disable */

usart_lin_mode_disable(USART0);

usart_lin_break_dection_length_config

The description of usart_lin_break_dection_length_config is shown as below:

Table 3-540. Function usart_lin_break_dection_length_config

Function name	usart_lin_break_dection_length_config
Function prototype	void usart_lin_break_dection_length_config(uint32_t usart_periph, uint32_t lblen);
Function descriptions	configure LIN break frame length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral

USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
lblen	two methods be used to enter or exit the mute mode
USART_LBLEN_10B	break frame length is 10 bits
USART_LBLEN_11B	break frame length is 11 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure LIN break frame length */

usart_lin_break_dection_length_config(USART0, USART_LBLEN_10B);

usart_send_break

The description of usart_send_break is shown as below:

Table 3-541. Function usart_send_break

Function name	usart_send_break
Function prototype	void usart_send_break(uint32_t usart_periph);
Function descriptions	send break frame
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 send break frame */

usart_send_break(USART0);

usart_halfduplex_enable

The description of usart_halfduplex_enable is shown as below:

Table 3-542. Function usart_halfduplex_enable

Function name	usart_halfduplex_enable
Function prototype	void usart_halfduplex_enable(uint32_t usart_periph);
Function descriptions	enable half duplex mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 half duplex mode*/
usart_halfduplex_enable(USART0);
```

usart_halfduplex_disable

The description of usart_halfduplex_disable is shown as below:

Table 3-543. Function usart_halfduplex_disable

Function name	usart_halfduplex_disable
Function prototype	void usart_halfduplex_disable(uint32_t usart_periph);
Function descriptions	disable half duplex mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 half duplex mode*/
usart_halfduplex_disable(USART0);
```

usart_synchronous_clock_enable

The description of usart_synchronous_clock_enable is shown as below:

Table 3-544. Function usart_synchronous_clock_enable

Function name	usart_synchronous_clock_enable
Function prototype	void usart_synchronous_clock_enable(uint32_t usart_periph);
Function descriptions	enable CK pin in synchronous mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 CK pin in synchronous mode */
usart_synchronous_clock_enable(USART0);
```

usart_synchronous_clock_disable

The description of usart_synchronous_clock_disable is shown as below:

Table 3-545. Function usart_synchronous_clock_disable

Function name	usart_synchronous_clock_disable
Function prototype	void usart_synchronous_clock_disable(uint32_t usart_periph);
Function descriptions	disable CK pin in synchronous mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 CK pin in synchronous mode */
usart_synchronous_clock_disable(USART0);
```

usart_synchronous_clock_config

The description of usart_synchronous_clock_config is shown as below:

Table 3-546. Function usart_synchronous_clock_config

Function name	usart_synchronous_clock_config
Function prototype	void usart_synchronous_clock_config(uint32_t usart_periph, uint32_t clen, uint32_t cph, uint32_t cpl);
Function descriptions	configure USART synchronous mode parameters
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
clen	CK length
USART_CLEN_NONE	there are 7 CK pulses for an 8 bit frame and 8 CK pulses for a 9 bit frame
USART_CLEN_EN	there are 8 CK pulses for an 8 bit frame and 9 CK pulses for a 9 bit frame
Input parameter{in}	
cph	clock phase
USART_CPH_1CK	first clock transition is the first data capture edge
USART_CPH_2CK	second clock transition is the first data capture edge
Input parameter{in}	
cpl	clock polarity
USART_CPL_LOW	steady low value on CK pin
USART_CPL_HIGH	steady high value on CK pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 synchronous mode parameters */

usart_synchronous_clock_config(USART0,USART_CLEN_EN,USART_CPH_2CK, USART_CPL_HIGH);

usart_guard_time_config

The description of usart_guard_time_config is shown as below:

Table 3-547. Function usart_guard_time_config

Function name	usart_guard_time_config
Function prototype	void usart_guard_time_config(uint32_t usart_periph, uint8_t gaut);
Function descriptions	configure guard time value in smartcard mode

Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
gaut	guard time value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 guard time value in smartcard mode */
usart_guard_time_config(USART0, 0x55);
```

usart_smartcard_mode_enable

The description of usart_smartcard_mode_enable is shown as below:

Table 3-548. Function usart_smartcard_mode_enable

Function name	usart_smartcard_mode_enable
Function prototype	void usart_smartcard_mode_enable(uint32_t usart_periph);
Function descriptions	enable smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 smartcard mode enable */
usart_smartcard_mode_enable(USART0);
```

usart_smartcard_mode_disable

The description of usart_smartcard_mode_disable is shown as below:

Table 3-549. Function usart_smartcard_mode_disable

Function name	usart_smartcard_mode_disable
Function prototype	void usart_smartcard_mode_disable(uint32_t usart_periph);
Function descriptions	disable smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 smartcard mode disable */
usart_smartcard_mode_disable(USART0);
```

usart_smartcard_mode_nack_enable

The description of usart_smartcard_mode_nack_enable is shown as below:

Table 3-550. Function usart_smartcard_mode_nack_enable

Function name	usart_smartcard_mode_nack_enable
Function prototype	void usart_smartcard_mode_nack_enable(uint32_t usart_periph);
Function descriptions	enable NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 NACK in smartcard mode */
usart_smartcard_mode_nack_enable(USART0);
```

usart_smartcard_mode_nack_disable

The description of usart_smartcard_mode_nack_disable is shown as below:

Table 3-551. Function usart_smartcard_mode_nack_disable

Function name	usart_smartcard_mode_nack_disable
Function prototype	void usart_smartcard_mode_nack_disable(uint32_t usart_periph);
Function descriptions	disable NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 NACK in smartcard mode */
usart_smartcard_mode_nack_disable(USART0);
```

usart_smartcard_autoretry_config

The description of usart_smartcard_autoretry_config is shown as below:

Table 3-552. Function usart_smartcard_autoretry_config

Function name	usart_smartcard_autoretry_config
Function prototype	void usart_smartcard_autoretry_config(uint32_t usart_periph, uint8_t scrtnum);
Function descriptions	configure smartcard auto-retry number
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
scrtnum	smartcard auto-retry number
0-0xFF	smartcard auto-retry number
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure smartcard auto-retry number */
usart_smartcard_autoretry_config (USART0, 0xFF);
```

usart_block_length_config

The description of usart_block_length_config is shown as below:

Table 3-553. Function usart_block_length_config

Function name	usart_block_length_config
Function prototype	void usart_block_length_config(uint32_t usart_periph, uint8_t bl);
Function descriptions	configure block length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
bl	block length
0-0xFF	block length
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure block length in Smartcard T=1 reception */
usart_block_length_config(USART0, 0xFF);
```

usart_irda_mode_enable

The description of usart_irda_mode_enable is shown as below:

Table 3-554. Function usart_irda_mode_enable

Function name	usart_irda_mode_enable
Function prototype	void usart_irda_mode_enable(uint32_t usart_periph);
Function descriptions	enable IrDA mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 IrDA mode */

usart_irda_mode_enable(USART0);
```

usart_irda_mode_disable

The description of usart_irda_mode_disable is shown as below:

Table 3-555. Function usart_irda_mode_disable

Function name	usart_irda_mode_disable
Function prototype	void usart_irda_mode_disable(uint32_t usart_periph);
Function descriptions	disable IrDA mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 IrDA mode */

usart_irda_mode_disable(USART0);
```

usart_prescaler_config

The description of usart_prescaler_config is shown as below:

Table 3-556. Function usart_prescaler_config

Function name	usart_prescaler_config
Function prototype	void usart_prescaler_config(uint32_t usart_periph, uint8_t psc);
Function descriptions	configure the peripheral clock prescaler in USART IrDA low-power mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
psc	0x00-0xFF

Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure the USART0 peripheral clock prescaler */

usart_prescaler_config(USART0, 0x00);

usart_irda_lowpower_config

The description of usart_irda_lowpower_config is shown as below:

Table 3-557. Function usart_irda_lowpower_config

Function name	usart_irda_lowpower_config
Function prototype	void usart_irda_lowpower_config(uint32_t usart_periph, uint32_t irlp);
Function descriptions	configure IrDA low-power
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
irlp	IrDA low-power or normal
USART_IRLP_LOW	low-power
USART_IRLP_NORMAL	normal
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* configure USART0 IrDA low-power */

usart_irda_lowpower_config(USART0, USART_IRLP_LOW);

usart_hardware_flow_rts_config

The description of usart_hardware_flow_rts_config is shown as below:

Table 3-558. Function usart_hardware_flow_rts_config

Function name	usart_hardware_flow_rts_config

Function prototype	void usart_hardware_flow_rts_config(uint32_t usart_periph, uint32_t rtsconfig);
Function descriptions	configure hardware flow control RTS
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
rtsconfig	enable or disable RTS
USART_RTS_ENABLE	enable RTS
USART_RTS_DISABLE	disable RTS
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 hardware flow control RTS */
usart_hardware_flow_cts_config(USART0, USART_RTS_ENABLE);
```

usart_hardware_flow_cts_config

The description of usart_hardware_flow_cts_config is shown as below:

Table 3-559. Function usart_hardware_flow_cts_config

Function name	usart_hardware_flow_cts_config
Function prototype	void usart_hardware_flow_cts_config(uint32_t usart_periph, uint32_t ctsconfig);
Function descriptions	configure hardware flow control RTS
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
ctsconfig	enable or disable CTS
USART_CTS_ENABLE	enable CTS
USART_CTS_DISABLE	disable CTS
Output parameter{out}	
-	-

Return value	
-	-

Example:

/* configure USART0 hardware flow control CTS */

usart_hardware_flow_cts_config(USART0, USART_CTS_ENABLE);

usart_dma_receive_config

The description of usart_dma_receive_config is shown as below:

Table 3-560. Function usart_dma_receive_config

Function name	usart_dma_receive_config
Function prototype	void usart_dma_receive_config(uint32_t usart_periph, uint32_t dmaconfig);
Function descriptions	configure USART DMA reception
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3
Input parameter{in}	
dmaconfig	USART DMA mode
USART_RECEIVE_DMA_ENABLE	enable USART DMA for reception
USART_RECEIVE_DMA_DISABLE	disable USART DMA for reception
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* USART0 DMA enable for reception */

usart_dma_receive_config(USART0, USART_RECEIVE_DMA_ENABLE);

usart_dma_transmit_config

The description of usart_dma_transmit_config is shown as below:

Table 3-561. Function usart_dma_transmit_config

Function name	usart_dma_transmit_config
Function prototype	void usart_dma_transmit_config(uint32_t usart_periph, uint32_t dmaconfig);

Function descriptions	configure USART DMA transmission
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3
Input parameter{in}	
dmaconfig	USART DMA mode
USART_TRANSMIT_DMA_ENABLE	enable USART DMA for transmission
USART_TRANSMIT_DMA_DISABLE	disable USART DMA for transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 DMA enable for transmission */
usart_dma_transmit_config(USART0, USART_TRANSMIT_DMA_ENABLE);
```

usart_hardware_flow_coherence_config

The description of usart_hardware_flow_coherence_config is shown as below:

Table 3-562. Function usart_hardware_flow_coherence_config

Function name	usart_hardware_flow_coherence_config
Function prototype	void usart_hardware_flow_coherence_config(uint32_t usart_periph, uint32_t hcm);
Function descriptions	configure hardware flow control coherence mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3
Input parameter{in}	
hcm	Hardware flow control coherence mode
USART_RTS_NONE_COHERENCE	nRTS signal equals to RBNE bit in USART_STAT0 register
USART_RTS_COHERENCE	nRTS signal is set when the last data bit has been sampled

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure hardware flow control coherence mode */
usart_hardware_flow_coherence_config(USART0, USART_RTS_COHERENCE);
```

usart_flag_get

The description of usart_flag_get is shown as below:

Table 3-563. Function usart_flag_get

Function name	usart_flag_get
Function prototype	FlagStatus usart_flag_get(uint32_t usart_periph, usart_flag_enum flag);
Function descriptions	get flag in STAT0/STAT1 register
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
flag	USART flags, refer to ***Table 3-514. usart_flag_enum***
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get flag USART0 state */
FlagStatus status;
status = usart_flag_get(USART0, USART_FLAG_TBE);
```

usart_flag_clear

The description of usart_flag_clear is shown as below:

Table 3-564. Function usart_flag_clear

Function name	usart_flag_clear
Function prototype	void usart_flag_clear(uint32_t usart_periph, usart_flag_enum flag);
Function descriptions	clear flag in STAT0/STAT1 register

Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
flag	USART flags, refer to *Table 3-514. usart_flag_enum*
USART_FLAG_CTSF	CTS change flag
USART_FLAG_LBDF	LIN break detected flag
USART_FLAG_TC	transmission complete
USART_FLAG_RBNE	read data buffer not empty
USART_FLAG_EB	end of block flag
USART_FLAG_RT	receiver timeout flag
USART_FLAG_EPERR	early parity error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear USART0 flag */
usart_flag_clear(USART0,USART_FLAG_TC);
```

usart_interrupt_enable

The description of usart_interrupt_enableis shown as below:

Table 3-565. Function usart_interrupt_enable

Function name	usart_interrupt_enable
Function prototype	void usart_interrupt_enable(uint32_t usart_periph, usart_interrupt_enum interrupt);
Function descriptions	enable USART interrupt
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
interrupt	USART interrupt, refer to *Table 3-516. usart_interrupt_enum*
USART_INT_PERR	parity error interrupt
USART_INT_TBE	transmitter buffer empty interrupt

USART_INT_TC	transmission complete interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt
USART_INT_IDLE	IDLE line detected interrupt
USART_INT_LBD	LIN break detected interrupt
USART_INT_ERR	error interrupt
USART_INT_CTS	CTS interrupt
USART_INT_RT	receive timeout event interrupt
USART_INT_EB	end of block event interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 TBE interrupt */
usart_interrupt_enable(USART0, USART_INT_TBE);
```

usart_interrupt_disable

The description of usart_interrupt_disable is shown as below:

Table 3-566. Function usart_interrupt_disable

Function name	usart_interrupt_disable
Function prototype	void usart_interrupt_disable(uint32_t usart_periph, usart_interrupt_enum interrupt);
Function descriptions	disable USART interrupt
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
int_flag	USART interrupt flag, refer to ***Table 3-516. usart_interrupt_enum***
USART_INT_PERR	parity error interrupt
USART_INT_TBE	transmitter buffer empty interrupt
USART_INT_TC	transmission complete interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt
USART_INT_IDLE	IDLE line detected interrupt
USART_INT_LBD	LIN break detected interrupt
USART_INT_ERR	error interrupt
USART_INT_CTS	CTS interrupt
USART_INT_RT	receive timeout event interrupt

USART_INT_EB	end of block event interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 TBE interrupt */

usart_interrupt_disable(USART0, USART_INT_TBE);
```

usart_interrupt_flag_get

The description of usart_interrupt_flag_get is shown as below:

Table 3-567. Function usart_interrupt_flag_get

Function name	usart_interrupt_flag_get
Function prototype	FlagStatus usart_interrupt_flag_get(uint32_t usart_periph, usart_interrupt_flag_enum int_flag);
Function descriptions	get USART interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
int_flag	USART interrupt flag, refer to ***Table 3-515. usart_interrupt_flag_enum***
USART_INT_FLAG_PERR	parity error interrupt and flag
USART_INT_FLAG_TBE	transmitter buffer empty interrupt and flag
USART_INT_FLAG_TC	transmission complete interrupt and flag
USART_INT_FLAG_RBNE	read data buffer not empty interrupt and flag
USART_INT_FLAG_RBNE_ORERR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_IDLE	IDLE line detected interrupt and flag
USART_INT_FLAG_LBD	LIN break detected interrupt and flag
USART_INT_FLAG_CTS	CTS interrupt and flag
USART_INT_FLAG_ER	error interrupt and overrun error

R_ORERR	
USART_INT_FLAG_ERR_NERR	error interrupt and noise error flag
USART_INT_FLAG_ERR_FERR	error interrupt and frame error flag
USART_INT_FLAG_EB	end of block event interrupt flag
USART_INT_FLAG_RT	receive timeout event interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the USART0 interrupt flag status */
FlagStatus status;
status = usart_interrupt_flag_get(USART0, USART_INT_FLAG_RBNE);
```

usart_interrupt_flag_clear

The description of usart_interrupt_flag_clear is shown as below:

Table 3-568. Function usart_interrupt_flag_clear

Function name	usart_interrupt_flag_clear
Function prototype	void usart_interrupt_flag_clear(uint32_t usart_periph, usart_interrupt_flag_enum int_flag);
Function descriptions	clear USART interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx/UARTx peripheral
USARTx	x=0,1,2
UARTx	x=3,4
Input parameter{in}	
int_flag	USART interrupt flag, refer to *Table 3-515. usart_interrupt_flag_enum*
USART_INT_FLAG_CTS	CTS change flag
USART_INT_FLAG_LBD	LIN break detected flag
USART_INT_FLAG_TC	transmission complete
USART_INT_FLAG_RBNE	read data buffer not empty
USART_INT_FLAG_EB	end of block event interrupt flag

USART_INT_FLAG_RT	receive timeout event interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the USART0 interrupt enable bit status */
usart_interrupt_flag_clear(USART0, USART_INT_FLAG_RBNE);
```

3.22. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions.The WWDGT registers are listed in chapter *3.22.1*, the FWDGT firmware functions are introduced in chapter *3.22.2.*

3.22.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

Table 3-569. WWDGT Registers

Registers	Descriptions
WWDGT_CTL	Control register
WWDGT_CFG	Configuration register
WWDGT_STAT	Status register

3.22.2. Descriptions of Peripheral functions

WWDGT firmware functions are listed in the table shown as below:

Table 3-570. WWDGT firmware function

Function name	Function description
wwdgt_deinit	reset the window watchdog timer configuration
wwdgt_enable	start the window watchdog timer counter
wwdgt_counter_update	configure the window watchdog timer counter value
wwdgt_config	configure counter value, window value, and prescaler divider value
wwdgt_interrupt_enable	enable early wakeup interrupt of WWDGT
wwdgt_flag_get	check early wakeup interrupt state of WWDGT
wwdgt_flag_clear	clear early wakeup interrupt state of WWDGT

wwdgt_deinit

The description of wwdgt_deinit is shown as below:

Table 3-571. Function wwdgt_deinit

Function name	wwdgt_deinit
Function prototype	void wwdgt_deinit(void);
Function descriptions	reset the window watchdog timer configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the window watchdog timer configuration */
wwdgt_deinit ( );
```

wwdgt_enable

The description of wwdgt_enable is shown as below:

Table 3-572. Function wwdgt_enable

Function name	wwdgt_enable
Function prototype	void wwdgt_enable (void);
Function descriptions	start the window watchdog timer counter
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start the window watchdog timer counter */
wwdgt_enable ( );
```

wwdgt_counter_update

The description of wwdgt_counter_update is shown as below:

Table 3-573. Function wwdgt_counter_update

Function name	wwdgt_counter_update
Function prototype	void wwdgt_counter_update(uint16_t counter_value);
Function descriptions	configure the window watchdog timer counter value
Precondition	-
The called functions	-
Input parameter{in}	
counter_value	0x00 - 0x7F
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* update WWDGT counter to 0x7F */
wwdgt_counter_update (127);
```

wwdgt_config

The description of wwdgt_config is shown as below:

Table 3-574. Function wwdgt_config

Function name	wwdgt_config
Function prototype	void wwdgt_config(uint16_t counter, uint16_t window, uint32_t prescaler);
Function descriptions	configure counter value, window value, and prescaler divider value
Precondition	-
The called functions	-
Input parameter{in}	
counter	0x00 - 0x7F
Input parameter{in}	
window	0x00 - 0x7F
Input parameter{in}	
prescaler	wwdgt prescaler value
WWDGT_CFG_PSC_DIV1	the time base of window watchdog counter = (PCLK1/4096)/1
WWDGT_CFG_PSC_DIV2	the time base of window watchdog counter = (PCLK1/4096)/2
WWDGT_CFG_PSC_DIV4	the time base of window watchdog counter = (PCLK1/4096)/4
WWDGT_CFG_PSC_D	the time base of window watchdog counter = (PCLK1/4096)/8

IV8	
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* confiure WWDGT counter value to 0x7F, window value to 0x50, prescaler divider value to 8 */

wwdgt_config (127, 80, WWDGT_CFG_PSC_DIV8);

wwdgt_interrupt_enable

The description of wwdgt_interrupt_enable is shown as below:

Table 3-575. Function wwdgt_interrupt_enable

Function name	wwdgt_interrupt_enable
Function prototype	void wwdgt_interrupt_enable(void);
Function descriptions	enable early wakeup interrupt of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable early wakeup interrupt of WWDGT */

wwdgt_interrupt_enable ( );

wwdgt_flag_get

The description of wwdgt_flag_get is shown as below:

Table 3-576. Function wwdgt_flag_get

Function name	wwdgt_flag_get
Function prototype	FlagStatus wwdgt_flag_get(void);
Function descriptions	check early wakeup interrupt state of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* test if the counter value update has reached the 0x40 */
FlagStatus status;
status = wwdgt_flag_get ( );
if(status == RESET)
{
...
}else
{
...
}
```

wwdgt_flag_clear

The description of wwdgt_flag_clear is shown as below:

Table 3-577. Function wwdgt_flag_clear

Function name	wwdgt_flag_clear
Function prototype	void wwdgt_flag_clear(void);
Function descriptions	clear early wakeup interrupt state of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear early wakeup interrupt state of WWDGT */
wwdgt_flag_clear( );
```

4. Revision history

Table 4-1. Revision history

Revison No.	Description	Date
1.0	Initial Release	Aug.15, 2022
1.1	1. Change Usart functions: usart_data_transmit usart_guard_time_config usart_smartcard_autoretry_config usart_block_length_config usart_dma_receive_config usart_dma_transmit_config 2. Change Timer decoder name	Dec.31, 2022
1.2	The entire DAC chapter has been modified	Dec.31, 2023
1.3	1. The parameter macro I2S_STD_PHILLIPS is modified to I2S_STD_PHILIPS 2. Update copyright information	Aug.8, 2025
1.4	1. Remove the functions ***spi_quad_io23_output_enable*** and ***spi_quad_io23_output_disable***.	Feb.4, 2026
1.5	1. Chapter ***Firmware Library Overview***. added GD32EBuilder_project, GD32EBuilder environment description.	Apr.14, 2026

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