

GigaDevice Semiconductor Inc.

GD32W51x

Arm[®] Cortex[®]-M33 32-bit MCU

**Firmware Library
User Guide**

Revision 1.2

(Dec. 2022)

Table of Contents

Table of Contents	2
List of Figures	5
List of Tables	6
1. Introduction	31
1.1. Rules of User Manual and Firmware Library	32
1.1.1. Peripherals.....	32
1.1.2. Naming rules.....	33
2. Firmware Library Overview	34
2.1. File Structure of Firmware Library	34
2.1.1. Examples Folder	35
2.1.2. Firmware Folder.....	35
2.1.3. Template Folder	36
2.1.4. Template_TrustZone Folder.....	39
2.1.5. Utilities Folder	44
2.2. File descriptions of Firmware Library	44
3. Firmware Library of Standard Peripherals	46
3.1. Overview of Firmware Library of Standard Peripherals.....	46
3.2. ADC	46
3.2.1. Descriptions of Peripheral registers.....	46
3.2.2. Descriptions of Peripheral functions	47
3.3. CAU	73
3.3.1. Descriptions of Peripheral registers.....	73
3.3.2. Descriptions of Peripheral functions	74
3.4. CRC	101
3.4.1. Descriptions of Peripheral registers.....	101
3.4.2. Descriptions of Peripheral functions	101
3.5. DBG	105
3.5.1. Descriptions of Peripheral registers.....	105
3.5.2. Descriptions of Peripheral functions	105
3.6. DCI.....	111
3.6.1. Descriptions of Peripheral registers.....	111
3.6.2. Descriptions of Peripheral functions	112
3.7. DMA.....	125
3.7.1. Descriptions of Peripheral registers.....	125
3.7.2. Descriptions of Peripheral functions	126

3.8. EFUSE	156
3.8.1. Descriptions of Peripheral registers	156
3.8.2. Descriptions of Peripheral functions	157
3.9. EXTI	170
3.9.1. Descriptions of Peripheral registers	170
3.9.2. Descriptions of Peripheral functions	171
3.10. FMC	181
3.10.1. Descriptions of Peripheral registers	181
3.10.2. Descriptions of Peripheral functions	182
3.11. FWDGT	207
3.11.1. Descriptions of Peripheral registers	207
3.11.2. Descriptions of Peripheral functions	207
3.12. GPIO	212
3.12.1. Descriptions of Peripheral registers	212
3.12.2. Descriptions of Peripheral functions	212
3.13. HAU	224
3.13.1. Descriptions of Peripheral registers	225
3.13.2. Descriptions of Peripheral functions	225
3.14. HPDF	244
3.14.1. Descriptions of Peripheral registers	244
3.14.2. Descriptions of Peripheral functions	244
3.15. I2C	300
3.15.1. Descriptions of Peripheral registers	301
3.15.2. Descriptions of Peripheral functions	301
3.16. ICACHE	340
3.16.1. Descriptions of Peripheral registers	340
3.16.2. Descriptions of Peripheral functions	340
3.17. MISC	351
3.17.1. Descriptions of Peripheral registers	351
3.17.2. Descriptions of Peripheral functions	352
3.18. PKCAU	359
3.18.1. Descriptions of Peripheral registers	359
3.18.2. Descriptions of Peripheral functions	360
3.19. PMU	388
3.19.1. Descriptions of Peripheral registers	388
3.19.2. Descriptions of Peripheral functions	389
3.20. QSPI	405
3.20.1. Descriptions of Peripheral registers	405
3.20.2. Descriptions of Peripheral functions	406

3.21.	RCU	423
3.21.1.	Descriptions of Peripheral registers	423
3.21.2.	Descriptions of Peripheral functions	424
3.22.	RTC	475
3.22.1.	Descriptions of Peripheral registers	475
3.22.2.	Descriptions of Peripheral functions	476
3.23.	SDIO	513
3.23.1.	Descriptions of Peripheral registers	513
3.23.2.	Descriptions of Peripheral functions	514
3.24.	SPI	544
3.24.1.	Descriptions of Peripheral registers	544
3.24.2.	Descriptions of Peripheral functions	544
3.25.	SQPI	572
3.25.1.	Descriptions of Peripheral registers	572
3.25.2.	Descriptions of Peripheral functions	572
3.26.	SYSCFG	578
3.26.1.	Descriptions of Peripheral registers	578
3.26.2.	Descriptions of Peripheral functions	579
3.27.	TIMER.....	592
3.27.1.	Descriptions of Peripheral registers	593
3.27.2.	Descriptions of Peripheral functions	593
3.28.	TRNG.....	649
3.28.1.	Descriptions of Peripheral registers	649
3.28.2.	Descriptions of Peripheral functions	649
3.29.	TSI	654
3.29.1.	Descriptions of Peripheral registers	655
3.29.2.	Descriptions of Peripheral functions	655
3.30.	TZPCU	675
3.30.1.	Descriptions of Peripheral registers	675
3.30.2.	Descriptions of Peripheral functions	676
3.31.	USART.....	698
3.31.1.	Descriptions of Peripheral registers	698
3.31.2.	Descriptions of Peripheral functions	699
3.32.	WWDGT.....	746
3.32.1.	Descriptions of Peripheral registers	746
3.32.2.	Descriptions of Peripheral functions	746
4.	Revision history	751

List of Figures

Figure 2-1. File structure of firmware library of GD32W51x.....	34
Figure 2-2. Select peripheral example files.....	37
Figure 2-3. Copy the peripheral example files	38
Figure 2-4. Open the project file.....	38
Figure 2-5. Configure project files	39
Figure 2-6. Compile-debug-download	39
Figure 2-7. Select peripheral example files.....	40
Figure 2-8. Copy the peripheral example files	41
Figure 2-9. Open the workspace	42
Figure 2-10. Configure project files	43
Figure 2-11. Compile-debug-download	44

List of Tables

Table 1-1. Peripherals.....	32
Table 2-1. Function descriptions of Firmware Library.....	44
Table 3-1. Peripheral function format of Firmware Library	46
Table 3-2. ADC Registers	46
Table 3-3. ADC firmware function	47
Table 3-4. Function adc_deinit	48
Table 3-5. Function adc_clock_config.....	49
Table 3-6. Function adc_enable	49
Table 3-7. Function adc_disable	50
Table 3-8. Function adc_dma_mode_enable	50
Table 3-9. Function adc_dma_mode_disable	51
Table 3-10. Function adc_dma_request_after_last_enable.....	51
Table 3-11. Function adc_dma_request_after_last_disable	52
Table 3-12. Function adc_discontinuous_mode_config.....	52
Table 3-13. Function adc_special_function_config	53
Table 3-14. Function adc_channel_9_to_11	54
Table 3-15. Function adc_data_alignment_config	55
Table 3-16. Function adc_channel_length_config	55
Table 3-17. Function adc_regular_channel_config	56
Table 3-18. Function adc_inserted_channel_config	57
Table 3-19. Function adc_inserted_channel_offset_config	58
Table 3-20. Function adc_external_trigger_config.....	58
Table 3-21. Function adc_external_trigger_source_config.....	59
Table 3-22. Function adc_software_trigger_enable	61
Table 3-23. Function adc_end_of_conversion_config.....	62
Table 3-24. Function adc_regular_data_read	62
Table 3-25. Function adc_inserted_data_read.....	63
Table 3-26. Function adc_watchdog_single_channel_enable	63
Table 3-27. Function adc_watchdog_group_channel_enable	64
Table 3-28. Function adc_watchdog_disable	65
Table 3-29. Function adc_watchdog_threshold_config.....	65
Table 3-30. Function adc_oversample_mode_config	66
Table 3-31. Function adc_oversample_mode_enable.....	67
Table 3-32. Function adc_oversample_mode_disable.....	68
Table 3-33. Function adc_flag_get	68
Table 3-34. Function adc_flag_clear	69
Table 3-35. Function adc_interrupt_enable	69
Table 3-36. Function adc_interrupt_disable	70
Table 3-37. Function adc_interrupt_flag_get	71
Table 3-38. Function adc_interrupt_flag_clear	71

Table 3-39. Function <code>adc_interrupt_flag_get</code>	72
Table 3-40. Function <code>adc_inserted_software_startconv_flag_get</code>	72
Table 3-41. CAU Registers	73
Table 3-42. CAU firmware function	74
Table 3-43. Structure <code>cau_key_parameter_struct</code>	75
Table 3-44. Structure <code>cau_iv_parameter_struct</code>	75
Table 3-45. Structure <code>cau_context_parameter_struct</code>	75
Table 3-46. Structure <code>cau_parameter_struct</code>	76
Table 3-47. Function <code>cau_deinit</code>	76
Table 3-48. Function <code>cau_struct_para_init</code>	76
Table 3-49. Function <code>cau_key_struct_para_init</code>	77
Table 3-50. Function <code>cau_iv_struct_para_init</code>	78
Table 3-51. Function <code>cau_context_struct_para_init</code>	78
Table 3-52. Function <code>cau_enable</code>	79
Table 3-53. Function <code>cau_disable</code>	79
Table 3-54. Function <code>cau_dma_enable</code>	80
Table 3-55. Function <code>cau_dma_disable</code>	80
Table 3-56. Function <code>cau_init</code>	81
Table 3-57. Function <code>cau_aes_keysize_config</code>	82
Table 3-58. Function <code>cau_key_init</code>	83
Table 3-59. Function <code>cau_iv_init</code>	83
Table 3-60. Function <code>cau_phase_config</code>	84
Table 3-61. Function <code>cau_fifo_flush</code>	84
Table 3-62. Function <code>cau_enable_state_get</code>	85
Table 3-63. Function <code>cau_data_write</code>	85
Table 3-64. Function <code>cau_data_read</code>	86
Table 3-65. Function <code>cau_context_save</code>	86
Table 3-66. Function <code>cau_context_restore</code>	87
Table 3-67. Function <code>cau_aes_ecb</code>	88
Table 3-68. Function <code>cau_aes_cbc</code>	89
Table 3-69. Function <code>cau_aes_ctr</code>	90
Table 3-70. Function <code>cau_aes_cfb</code>	91
Table 3-71. Function <code>cau_aes_ofb</code>	92
Table 3-72. Function <code>cau_aes_gcm</code>	92
Table 3-73. Function <code>cau_aes_ccm</code>	94
Table 3-74. Function <code>cau_tdes_ecb</code>	95
Table 3-75. Function <code>cau_tdes_cbc</code>	96
Table 3-76. Function <code>cau_des_ecb</code>	96
Table 3-77. Function <code>cau_des_cbc</code>	97
Table 3-78. Function <code>cau_interrupt_enable</code>	98
Table 3-79. Function <code>cau_interrupt_disable</code>	99
Table 3-80. Function <code>cau_interrupt_flag_get</code>	99
Table 3-81. Function <code>cau_flag_get</code>	100
Table 3-82. CRC Registers	101

Table 3-83. CRC firmware function	101
Table 3-84. Function crc_deinit	101
Table 3-85. Function crc_data_register_reset	102
Table 3-86. Function crc_data_register_read	102
Table 3-87. Function crc_free_data_register_read.....	103
Table 3-88. Function crc_free_data_register_write.....	103
Table 3-89. Function crc_single_data_calculate	104
Table 3-90. Function crc_block_data_calculate	104
Table 3-91. DBG Registers.....	105
Table 3-92. DBG firmware function	105
Table 3-93. Enum dbg_periph_enum	106
Table 3-94. Function dbg_deinit.....	106
Table 3-95. Function dbg_id_get.....	107
Table 3-96. Function dbg_low_power_enable	107
Table 3-97. Function dbg_low_power_disable	108
Table 3-98. Function dbg_periph_enable	108
Table 3-99. Function dbg_periph_disable	109
Table 3-100. Function dbg_trace_pin_enable	110
Table 3-101. Function dbg_trace_pin_disable	110
Table 3-102. Function dbg_trace_pin_mode_set	111
Table 3-103. DCI Registers.....	111
Table 3-104. DCI firmware function	112
Table 3-105. Structure dci_parameter_struct	113
Table 3-106. Function dci_deinit	113
Table 3-107. Function dci_struct_para_init.....	114
Table 3-108. Function dci_init	114
Table 3-109. Function dci_enable	115
Table 3-110. Function dci_disable.....	115
Table 3-111. Function dci_capture_enable.....	116
Table 3-112. Function dci_capture_disable	116
Table 3-113. Function dci_jpeg_enable	117
Table 3-114. Function dci_jpeg_disable	117
Table 3-115. Function dci_crop_window_enable	118
Table 3-116. Function dci_crop_window_disable	118
Table 3-117. Function dci_crop_window_config	119
Table 3-118. Function dci_embedded_sync_enable	119
Table 3-119. Function dci_embedded_sync_disable	120
Table 3-120. Function dci_sync_codes_config	120
Table 3-121. Function dci_sync_codes_unmask_config.....	121
Table 3-122. Function dci_data_read	121
Table 3-123. Function dci_flag_get	122
Table 3-124. Function dci_interrupt_enable.....	123
Table 3-125. Function dci_interrupt_disable.....	123
Table 3-126. Function dci_interrupt_flag_get	124

Table 3-127. Function dci_interrupt_flag_clear	124
Table 3-128. DMA Registers	125
Table 3-129. DMA firmware function	126
Table 3-130. Enum dma_channel_enum	127
Table 3-131. Enum dma_subperipheral_enum	127
Table 3-132. Structure dma_multi_data_parameter_struct	128
Table 3-133. Structure dma_single_data_parameter_struct	128
Table 3-134. Function dma_deinit	128
Table 3-135. Function dma_single_data_para_struct_init	129
Table 3-136. Function dma_multi_data_para_struct_init	130
Table 3-137. Function dma_single_data_mode_init	130
Table 3-138. Function dma_multi_data_mode_init	131
Table 3-139. Function dma_periph_address_config	132
Table 3-140. Function dma_memory_address_config	133
Table 3-141. Function dma_transfer_number_config	134
Table 3-142. Function dma_transfer_number_get	134
Table 3-143. Function dma_priority_config	135
Table 3-144. Function dma_memory_burst_beats_config	136
Table 3-145. Function dma_periph_burst_beats_config	136
Table 3-146. Function dma_memory_width_config	137
Table 3-147. Function dma_periph_width_config	138
Table 3-148. Function dma_memory_address_generation_config	139
Table 3-149. Function dma_peripheral_address_generation_config	140
Table 3-150. Function dma_circulation_enable	140
Table 3-151. Function dma_circulation_disable	141
Table 3-152. Function dma_channel_enable	142
Table 3-153. Function dma_channel_disable	142
Table 3-154. Function dma_transfer_direction_config	143
Table 3-155. Function dma_switch_buffer_mode_config	144
Table 3-156. Function dma_using_memory_get	144
Table 3-157. Function dma_channel_subperipheral_select	145
Table 3-158. Function dma_flow_controller_config	146
Table 3-159. Function dma_switch_buffer_mode_enable	147
Table 3-160. Function dma_switch_buffer_mode_disable	147
Table 3-161. Function dma_fifo_status_get	148
Table 3-162. Function dma_flag_get	148
Table 3-163. Function dma_flag_clear	149
Table 3-164. Function dma_interrupt_flag_get	150
Table 3-165. Function dma_interrupt_flag_clear	151
Table 3-166. Function dma_interrupt_enable	152
Table 3-167. Function dma_interrupt_disable	152
Table 3-168. Function dma_security_config	153
Table 3-169. Function dma_priv_config	154
Table 3-170. Function dma_illegal_access_flag_get	155

Table 3-171. Function dma_illegal_access_flag_clear	155
Table 3-172. EFUSE Registers	156
Table 3-173. EFUSE firmware function	157
Table 3-174. Enum efuse_flag_enum	158
Table 3-175. Enum efuse_clear_flag_enum	158
Table 3-176. Enum efuse_int_enum	158
Table 3-177. Enum efuse_int_flag_enum	158
Table 3-178. efuse_clear_int_flag_enum	158
Table 3-179. Enum efuse_tz_enum	158
Table 3-180. Function efuse_read	159
Table 3-181. Function efuse_write	159
Table 3-182. Function efuse_boot_config	160
Table 3-183. Function efuse_tz_control_config	160
Table 3-184. Function efuse_fp_config	161
Table 3-185. Function efuse_mcu_init_data_write	162
Table 3-186. Function efuse_aes_key_write	162
Table 3-187. Function efuse_rotpk_key_write	163
Table 3-188. Function efuse_dp_write	163
Table 3-189. Function efuse_iak_write	164
Table 3-190. Function efuse_user_data_write	165
Table 3-191. Function efuse_software_trustzone_enable	165
Table 3-192. Function efuse_software_trustzone_disable	166
Table 3-193. Function efuse_boot_address_get	166
Table 3-194. Function efuse_flag_get	167
Table 3-195. Function efuse_flag_clear	168
Table 3-196. Function efuse_interrupt_enable	168
Table 3-197. Function efuse_interrupt_disable	169
Table 3-198. Function efuse_interrupt_flag_get	169
Table 3-199. Function efuse_interrupt_flag_clear	170
Table 3-200. EXTI Registers	171
Table 3-201. EXTI firmware function	171
Table 3-202. Enum exti_line_enum	171
Table 3-203. Enum exti_mode_enum	172
Table 3-204. Enum exti_trig_type_enum	172
Table 3-205. Function exti_deinit	173
Table 3-206. Function exti_init	173
Table 3-207. Function exti_interrupt_enable	174
Table 3-208. Function exti_interrupt_disable	174
Table 3-209. Function exti_event_enable	175
Table 3-210. Function exti_event_disable	175
Table 3-211. Function exti_security_enable	176
Table 3-212. Function exti_security_disable	176
Table 3-213. Function exti_privilege_enable	177
Table 3-214. Function exti_privilege_disable	177

Table 3-215. Function exti_lock_enable	178
Table 3-216. Function exti_software_interrupt_enable.....	178
Table 3-217. Function exti_software_interrupt_disable.....	179
Table 3-218. Function exti_flag_get.....	179
Table 3-219. Function exti_flag_clear	180
Table 3-220. Function exti_interrupt_flag_get	180
Table 3-221. Function exti_interrupt_flag_clear	181
Table 3-222. FMC Registers	182
Table 3-223. FMC firmware function	182
Table 3-224. fmc_state_enum.....	184
Table 3-225. Function fmc_unlock.....	184
Table 3-226. Function fmc_lock	184
Table 3-227. Function fmc_page_erase.....	185
Table 3-228. Function fmc_mass_erase.....	185
Table 3-229. Function fmc_word_program	186
Table 3-230. Function fmc_continuous_program	187
Table 3-231. Function sram1_reset_enable	187
Table 3-232. Function sram1_reset_disable	188
Table 3-233. Function fmc_privilege_enable	188
Table 3-234. Function fmc_privilege_disable	189
Table 3-235. Function ob_unlock.....	189
Table 3-236. Function ob_lock	190
Table 3-237. Function ob_start.....	190
Table 3-238. Function ob_reload.....	191
Table 3-239. Function ob_security_protection_config	191
Table 3-240. Function ob_trustzone_enable	192
Table 3-241. Function ob_trustzone_disable	192
Table 3-242. Function ob_user_write.....	193
Table 3-243. Function ob_write_protection_config.....	194
Table 3-244. Function ob_secmark_config	194
Table 3-245. Function ob_dmp_access_enable	195
Table 3-246. Function ob_dmp_access_disable	196
Table 3-247. Function ob_dmp_config	196
Table 3-248. Function ob_dmp_enable.....	197
Table 3-249. Function ob_dmp_disable.....	197
Table 3-250. Function fmc_no_rtdec_config	198
Table 3-251. Function fmc_offset_region_config.....	199
Table 3-252. Function fmc_offset_value_config.....	199
Table 3-253. Function ob_write_protection_get	200
Table 3-254. Function ob_user_get.....	200
Table 3-255. Function ob_security_protection_flag_get	201
Table 3-256. Function ob_trustzone_state_get	201
Table 3-257. Function ob_memory_mode_state_get	202
Table 3-258. Function ob_exist_state_get.....	202

Table 3-259. Function <code>fmc_flag_get</code>	203
Table 3-260. Function <code>fmc_flag_clear</code>	204
Table 3-261. Function <code>fmc_interrupt_enable</code>	204
Table 3-262. Function <code>fmc_interrupt_disable</code>	205
Table 3-263. Function <code>fmc_interrupt_flag_get</code>	205
Table 3-264. Function <code>fmc_interrupt_flag_clear</code>	206
Table 3-265. FWDGT Registers.....	207
Table 3-266. FWDGT firmware function.....	207
Table 3-267. Function <code>fwdgt_write_enable</code>	207
Table 3-268. Function <code>fwdgt_write_disable</code>	208
Table 3-269. Function <code>fwdgt_enable</code>	208
Table 3-270. Function <code>fwdgt_prescaler_value_config</code>	209
Table 3-271. Function <code>fwdgt_reload_value_config</code>	209
Table 3-272. Function <code>fwdgt_counter_reload</code>	210
Table 3-273. Function <code>fwdgt_config</code>	210
Table 3-274. Function <code>fwdgt_flag_get</code>	211
Table 3-275. GPIO Registers	212
Table 3-276. GPIO firmware function	212
Table 3-277. Function <code>gpio_deinit</code>	213
Table 3-278. Function <code>gpio_mode_set</code>	213
Table 3-279. Function <code>gpio_output_options_set</code>	214
Table 3-280. Function <code>gpio_bit_set</code>	215
Table 3-281. Function <code>gpio_bit_reset</code>	216
Table 3-282. Function <code>gpio_bit_write</code>	216
Table 3-283. Function <code>gpio_port_write</code>	217
Table 3-284. Function <code>gpio_input_bit_get</code>	218
Table 3-285. Function <code>gpio_input_port_get</code>	218
Table 3-286. Function <code>gpio_output_bit_get</code>	219
Table 3-287. Function <code>gpio_output_port_get</code>	219
Table 3-288. Function <code>gpio_af_set</code>	220
Table 3-289. Function <code>gpio_pin_lock</code>	221
Table 3-290. Function <code>gpio_bit_toggle</code>	222
Table 3-291. Function <code>gpio_port_toggle</code>	222
Table 3-292. Function <code>gpio_bit_set_sec_cfg</code>	223
Table 3-293. Function <code>gpio_bit_reset_sec_cfg</code>	223
Table 3-294. Function <code>gpio_sec_cfg_bit_get</code>	224
Table 3-295. HAU Registers	225
Table 3-296. HAU firmware function	225
Table 3-297. Structure <code>hau_init_parameter_struct</code>	226
Table 3-298. Structure <code>hau_digest_parameter_struct</code>	226
Table 3-299. Structure <code>hau_context_parameter_struct</code>	226
Table 3-300. Function <code>hau_deinit</code>	227
Table 3-301. Function <code>hau_init</code>	227
Table 3-302. Function <code>hau_init_struct_para_init</code>	228

Table 3-303. Function hau_reset	229
Table 3-304. Function hau_last_word_validbits_num_config.....	229
Table 3-305. Function hau_data_write.....	230
Table 3-306. Function hau_infifo_words_num_get.....	230
Table 3-307. Function hau_digest_read	231
Table 3-308. Function hau_digest_calculation_enable.....	231
Table 3-309. Function hau_multiple_single_dma_config.....	232
Table 3-310. Function hau_dma_enable.....	232
Table 3-311. Function hau_dma_disable.....	233
Table 3-312. Function hau_context_struct_para_init.....	233
Table 3-313. Function hau_context_save.....	234
Table 3-314. Function hau_context_restore	234
Table 3-315. Function hau_hash_sha_1	235
Table 3-316. Function hau_hmac_sha_1	235
Table 3-317. Function hau_hash_sha_224.....	236
Table 3-318. Function hau_hmac_sha_224.....	237
Table 3-319. Function hau_hash_sha_256.....	237
Table 3-320. Function hau_hmac_sha_256.....	238
Table 3-321. Function hau_hash_md5.....	238
Table 3-322. Function hau_hmac_md5.....	239
Table 3-323. Function hau_flag_get.....	240
Table 3-324. Function hau_flag_clear.....	240
Table 3-325. Function hau_interrupt_enable	241
Table 3-326. Function hau_interrupt_disable	241
Table 3-327. Function hau_interrupt_flag_get.....	242
Table 3-328. Function hau_interrupt_flag_clear.....	243
Table 3-329. HPDF Registers	244
Table 3-330. HPDF firmware function	244
Table 3-331. Enum hpdf_channel_enum	247
Table 3-332. Enum hpdf_filter_enum	247
Table 3-333. Enum hpdf_flag_enum	247
Table 3-334. Enum hpdf_interrput_flag_enum	248
Table 3-335. Enum hpdf_interrput_enum.....	249
Table 3-336. Structure hpdf_channel_parameter_struct	249
Table 3-337. Structure hpdf_filter_parameter_struct.....	249
Table 3-338. Structure hpdf_rc_parameter_struct	250
Table 3-339. Structure hpdf_ic_parameter_struct.....	250
Table 3-340. Function hpdf_deinit.....	250
Table 3-341. Function hpdf_channel_struct_para_init	251
Table 3-342. Function hpdf_filter_struct_para_init	251
Table 3-343. Function hpdf_rc_struct_para_init.....	252
Table 3-344. Function hpdf_ic_struct_para_init	252
Table 3-345. Function hpdf_enable.....	253
Table 3-346. Function hpdf_disable.....	253

Table 3-347. Function <code>hpdf_channel_init</code>	254
Table 3-348. Function <code>hpdf_filter_init</code>	255
Table 3-349. Function <code>hpdf_rc_init</code>	256
Table 3-350. Function <code>hpdf_ic_init</code>	257
Table 3-351. Function <code>hpdf_clock_output_config</code>	257
Table 3-352. Function <code>hpdf_clock_output_source_config</code>	258
Table 3-353. Function <code>hpdf_clock_output_duty_mode_disable</code>	259
Table 3-354. Function <code>hpdf_clock_output_duty_mode_enable</code>	259
Table 3-355. Function <code>hpdf_clock_output_divider_config</code>	259
Table 3-356. Function <code>hpdf_channel_enable</code>	260
Table 3-357. Function <code>hpdf_channel_disable</code>	260
Table 3-358. Function <code>hpdf_spi_clock_source_config</code>	261
Table 3-359. Function <code>hpdf_serial_interface_type_config</code>	262
Table 3-360. Function <code>hpdf_malfunction_monitor_disable</code>	262
Table 3-361. Function <code>hpdf_malfunction_monitor_enable</code>	263
Table 3-362. Function <code>hpdf_clock_loss_disable</code>	263
Table 3-363. Function <code>hpdf_clock_loss_enable</code>	264
Table 3-364. Function <code>hpdf_channel_pin_redirection_disable</code>	264
Table 3-365. Function <code>hpdf_channel_pin_redirection_enable</code>	265
Table 3-366. Function <code>hpdf_channel_multiplexer_config</code>	265
Table 3-367. Function <code>hpdf_data_pack_mode_config</code>	266
Table 3-368. Function <code>hpdf_data_right_bit_shift_config</code>	267
Table 3-369. Function <code>hpdf_calibration_offset_config</code>	267
Table 3-370. Function <code>hpdf_malfunction_break_signal_config</code>	268
Table 3-371. Function <code>hpdf_malfunction_counter_config</code>	268
Table 3-372. Function <code>hpdf_write_parallel_data_standard_mode</code>	269
Table 3-373. Function <code>hpdf_write_parallel_data_interleaved_mode</code>	270
Table 3-374. Function <code>hpdf_write_parallel_data_dual_mode</code>	270
Table 3-375. Function <code>hpdf_pulse_skip_update</code>	271
Table 3-376. Function <code>hpdf_pulse_skip_read</code>	271
Table 3-377. Function <code>hpdf_filter_enable</code>	272
Table 3-378. Function <code>hpdf_filter_disable</code>	272
Table 3-379. Function <code>hpdf_filter_config</code>	273
Table 3-380. Function <code>hpdf_integrator_oversample</code>	274
Table 3-381. Function <code>hpdf_threshold_monitor_filter_config</code>	274
Table 3-382. Function <code>hpdf_threshold_monitor_filter_read_data</code>	275
Table 3-383. Function <code>hpdf_threshold_monitor_fast_mode_disable</code>	275
Table 3-384. Function <code>hpdf_threshold_monitor_fast_mode_enable</code>	276
Table 3-385. Function <code>hpdf_threshold_monitor_channel</code>	276
Table 3-386. Function <code>hpdf_threshold_monitor_high_threshold</code>	277
Table 3-387. Function <code>hpdf_threshold_monitor_low_threshold</code>	278
Table 3-388. Function <code>hpdf_high_threshold_break_signal</code>	278
Table 3-389. Function <code>hpdf_low_threshold_break_signal</code>	279
Table 3-390. Function <code>hpdf_extremes_monitor_channel</code>	280

Table 3-391. Function <code>hpdf_extremes_monitor_maximum_get</code>	280
Table 3-392. Function <code>hpdf_extremes_monitor_minimum_get</code>	281
Table 3-393. Function <code>hpdf_rc_continuous_disable</code>	281
Table 3-394. Function <code>hpdf_rc_continuous_enable</code>	282
Table 3-395. Function <code>hpdf_rc_start_by_software</code>	282
Table 3-396. Function <code>hpdf_rc_syn_disable</code>	283
Table 3-397. Function <code>hpdf_rc_syn_enable</code>	283
Table 3-398. Function <code>hpdf_rc_dma_disable</code>	284
Table 3-399. Function <code>hpdf_rc_dma_enable</code>	284
Table 3-400. Function <code>hpdf_rc_channel_config</code>	285
Table 3-401. Function <code>hpdf_rc_fast_mode_disable</code>	286
Table 3-402. Function <code>hpdf_rc_fast_mode_enable</code>	286
Table 3-403. Function <code>hpdf_rc_data_get</code>	287
Table 3-404. Function <code>hpdf_rc_channel_get</code>	287
Table 3-405. Function <code>hpdf_ic_start_by_software</code>	288
Table 3-406. Function <code>hpdf_ic_syn_disable</code>	288
Table 3-407. Function <code>hpdf_ic_syn_enable</code>	289
Table 3-408. Function <code>hpdf_ic_dma_disable</code>	289
Table 3-409. Function <code>hpdf_ic_dma_enable</code>	290
Table 3-410. Function <code>hpdf_ic_scan_mode_disable</code>	290
Table 3-411. Function <code>hpdf_ic_scan_mode_enable</code>	291
Table 3-412. Function <code>hpdf_ic_trigger_signal_disable</code>	291
Table 3-413. Function <code>hpdf_ic_trigger_signal_config</code>	292
Table 3-414. Function <code>hpdf_ic_channel_config</code>	292
Table 3-415. Function <code>hpdf_ic_data_get</code>	293
Table 3-416. Function <code>hpdf_ic_channel_get</code>	294
Table 3-417. Function <code>hpdf_flag_get</code>	294
Table 3-418. Function <code>hpdf_flag_clear</code>	295
Table 3-419. Function <code>hpdf_interrupt_enable</code>	297
Table 3-420. Function <code>hpdf_interrupt_disable</code>	298
Table 3-421. Function <code>hpdf_interrupt_flag_get</code>	299
Table 3-422. Function <code>hpdf_interrupt_flag_clear</code>	300
Table 3-423. I2C Registers	301
Table 3-424. I2C firmware function	301
Table 3-425. <code>i2c_interrupt_flag_enum</code>	303
Table 3-426. Function <code>i2c_deinit</code>	303
Table 3-427. Function <code>i2c_timing_config</code>	304
Table 3-428. Function <code>i2c_digital_noise_filter_config</code>	305
Table 3-429. Function <code>i2c_analog_noise_filter_enable</code>	306
Table 3-430. Function <code>i2c_analog_noise_filter_disable</code>	306
Table 3-431. Function <code>i2c_master_clock_config</code>	307
Table 3-432. Function <code>i2c_master_addressing</code>	307
Table 3-433. Function <code>i2c_address10_header_enable</code>	308
Table 3-434. Function <code>i2c_address10_header_disable</code>	308

Table 3-435. Function i2c_address10_enable.....	309
Table 3-436. Function i2c_address10_disable.....	309
Table 3-437. Function i2c_automatic_end_enable	310
Table 3-438. Function i2c_automatic_end_disable	310
Table 3-439. Function i2c_slave_response_to_gcall_enable.....	311
Table 3-440. Function i2c_slave_response_to_gcall_disable.....	311
Table 3-441. Function i2c_stretch_scl_low_enable	312
Table 3-442. Function i2c_stretch_scl_low_disable	312
Table 3-443. Function i2c_address_config	313
Table 3-444. Function i2c_address_bit_compare_config.....	314
Table 3-445. Function i2c_address_disable.....	315
Table 3-446. Function i2c_second_address_config.....	315
Table 3-447. Function i2c_second_address_disable	316
Table 3-448. Function i2c_receved_address_get	317
Table 3-449. Function i2c_slave_byte_control_enable.....	317
Table 3-450. Function i2c_slave_byte_control_disable.....	318
Table 3-451. Function i2c_nack_enable	318
Table 3-452. Function i2c_nack_disable	319
Table 3-453. Function i2c_wakeup_from_deepsleep_enable	319
Table 3-454. Function i2c_wakeup_from_deepsleep_disable	320
Table 3-455. Function i2c_enable.....	320
Table 3-456. Function i2c_disable.....	321
Table 3-457. Function i2c_start_on_bus	321
Table 3-458. Function i2c_stop_on_bus.....	322
Table 3-459. Function i2c_data_transmit	322
Table 3-460. Function i2c_data_receive	323
Table 3-461. Function i2c_reload_enable.....	323
Table 3-462. Function i2c_reload_disable.....	324
Table 3-463. Function i2c_transfer_byte_number_config.....	324
Table 3-464. Function i2c_dma_enable	325
Table 3-465. Function i2c_dma_disable	325
Table 3-466. Function i2c_pec_transfer	326
Table 3-467. Function i2c_pec_enable	326
Table 3-468. Function i2c_pec_disable	327
Table 3-469. Function i2c_pec_value_get	327
Table 3-470. Function i2c_smbus_alert_enable	328
Table 3-471. Function i2c_smbus_alert_disable	329
Table 3-472. Function i2c_smbus_default_addr_enable	329
Table 3-473. Function i2c_smbus_default_addr_disable	330
Table 3-474. Function i2c_smbus_host_addr_enable	330
Table 3-475. Function i2c_smbus_host_addr_disable	331
Table 3-476. Function i2c_extented_clock_timeout_enable	331
Table 3-477. Function i2c_extented_clock_timeout_disable	332
Table 3-478. Function i2c_clock_timeout_enable	332

Table 3-479. Function i2c_clock_timeout_disable	333
Table 3-480. Function i2c_bus_timeout_b_config	333
Table 3-481. Function i2c_bus_timeout_a_config.....	334
Table 3-482. Function i2c_idle_clock_timeout_config.....	334
Table 3-483. Function i2c_flag_get	335
Table 3-484. Function i2c_flag_clear	336
Table 3-485. Function i2c_interrupt_enable.....	337
Table 3-486. Function i2c_interrupt_disable.....	337
Table 3-487. Function i2c_interrupt_flag_get	338
Table 3-488. Function i2c_interrupt_flag_clear	339
Table 3-489. ICAHCE Registers	340
Table 3-490. ICACHE firmware function	341
Table 3-491. Structure icache_remap_struct	341
Table 3-492. Function icache_enable	341
Table 3-493. Function icache_disable	342
Table 3-494. Function icache_monitor_enable	342
Table 3-495. Function icache_monitor_disable	343
Table 3-496. Function icache_monitor_reset.....	344
Table 3-497. Function icache_way_configure	344
Table 3-498. Function icache_burst_type_select.....	345
Table 3-499. Function icache_invalidation.....	345
Table 3-500. Function icache_hitvalue_get.....	346
Table 3-501. Function icache_missvalue_get.....	346
Table 3-502. Function icache_remap_enable	347
Table 3-503. Function icache_remap_disable	347
Table 3-504. Function icache_flag_get	348
Table 3-505. Function icache_flag_clear	348
Table 3-506. Function icache_interrupt_enable	349
Table 3-507. Function icache_interrupt_disable	350
Table 3-508. Function icache_flag_get	350
Table 3-509. Function icache_interrupt_flag_clear	351
Table 3-510. NVIC Registers	351
Table 3-511. SysTick Registers	352
Table 3-512. MISC firmware function	352
Table 3-513. IRQn_Type.....	352
Table 3-514. Function nvic_priority_group_set.....	354
Table 3-515. Function nvic_irq_enable.....	355
Table 3-516. Function nvic_irq_disable.....	356
Table 3-517. Function nvic_system_reset.....	356
Table 3-518. Function nvic_vector_table_set	357
Table 3-519. Function system_lowpower_set.....	357
Table 3-520. Function system_lowpower_reset	358
Table 3-521. Function systick_clksource_set.....	359
Table 3-522. PKCAU Registers	359

Table 3-523. PKCAU firmware function	360
Table 3-524. Structure pkcau_mont_parameter_struct	361
Table 3-525. Structure pkcau_mod_parameter_struct.....	361
Table 3-526. Structure pkcau_mod_exp_parameter_struct	361
Table 3-527. Structure pkcau_mod_inver_parameter_struct.....	361
Table 3-528. Structure pkcau_mod_reduc_parameter_struct.....	361
Table 3-529. Structure pkcau_arithmetic_parameter_struct	362
Table 3-530. Structure pkcau crt_parameter_struct.....	362
Table 3-531. Structure pkcau_ec_group_parameter_struct	362
Table 3-532. Structure pkcau_point_parameter_struct	363
Table 3-533. Structure pkcau_signature_parameter_struct.....	363
Table 3-534. Structure pkcau_hash_parameter_struct.....	363
Table 3-535. Function pkcau_deinit	363
Table 3-536. Function pkcau_mont_struct_para_init.....	364
Table 3-537. Function pkcau_mod_struct_para_init.....	364
Table 3-538. Function pkcau_mod_exp_struct_para_init.....	365
Table 3-539. Function pkcau_mod_inver_struct_para_init	365
Table 3-540. Function pkcau_arithmetic_struct_para_init	366
Table 3-541. Function pkcau crt_struct_para_init.....	366
Table 3-542. Function pkcau_ec_group_struct_para_init	367
Table 3-543. Function pkcau_point_struct_para_init.....	367
Table 3-544. Function pkcau_signature_struct_para_init	368
Table 3-545. Function pkcau_hash_struct_para_init	369
Table 3-546. Function pkcau_mod_reduc_struct_para_init	369
Table 3-547. Function pkcau_enable	370
Table 3-548. Function pkcau_disable	370
Table 3-549. Function pkcau_start.....	371
Table 3-550. Function pkcau_mode_set.....	371
Table 3-551. Function pkcau_mont_param_operation	372
Table 3-552. Function pkcau_mod_operation	373
Table 3-553. Function pkcau_mod_exp_operation	374
Table 3-554. Function pkcau_mod_inver_operation.....	375
Table 3-555. Function pkcau_mod_reduc_operation.....	376
Table 3-556. Function pkcau_arithmetic_operation.....	377
Table 3-557. Function pkcau crt_exp_operation	378
Table 3-558. Function pkcau_point_check_operation	379
Table 3-559. Function pkcau_point_mul_operation.....	380
Table 3-560. Function pkcau_ecdsa_sign_operation	381
Table 3-561. Function pkcau_ecdsa_verification_operation	382
Table 3-562. Function pkcau_memread.....	384
Table 3-563. Function pkcau_flag_get.....	385
Table 3-564. Function pkcau_flag_clear.....	385
Table 3-565. Function pkcau_interrupt_enable	386
Table 3-566. Function pkcau_interrupt_disable	387

Table 3-567. Function pkcau_interrupt_flag_get	387
Table 3-568. Function pkcau_interrupt_flag_clear	388
Table 3-569. PMU Registers	389
Table 3-570. PMU firmware function	389
Table 3-571. Function pmu_deinit	390
Table 3-572. Function pmu_lvd_select	390
Table 3-573. Function pmu_lvd_disable	391
Table 3-574. Function pmu_vlvd_enable	391
Table 3-575. Function pmu_vlvd_disable	392
Table 3-576. Function pmu_ldo_output_select	392
Table 3-577. Function pmu_to_sleepmode	393
Table 3-578. Function pmu_to_deepsleepmode	393
Table 3-579. Function pmu_to_standbymode	394
Table 3-580. Function pmu_wakeup_pin_enable	395
Table 3-581. Function pmu_wakeup_pin_disable	395
Table 3-582. Function pmu_backup_write_enable	396
Table 3-583. Function pmu_backup_write_disable	396
Table 3-584. Function pmu_wifi_power_enable	397
Table 3-585. Function pmu_wifi_power_disable	397
Table 3-586. Function pmu_wifi_sram_control	398
Table 3-587. Function pmu_rf_force_enable	398
Table 3-588. Function pmu_rf_force_disable	399
Table 3-589. Function pmu_rf_sequence_config	399
Table 3-590. Function pmu_security_enable	400
Table 3-591. Function pmu_security_disable	401
Table 3-592. Function pmu_privilege_enable	402
Table 3-593. Function pmu_privilege_disable	402
Table 3-594. Function pmu_flag_reset	403
Table 3-595. Function pmu_flag_get	403
Table 3-596. QSPI registers	405
Table 3-597. QSPI firmware function	406
Table 3-598. qspi_init_struct	406
Table 3-599. qspi_command_struct	407
Table 3-600. qspi_autopolling_struct	408
Table 3-601. Function qspi_deinit	408
Table 3-602. Function qspi_init_struct_para_init	408
Table 3-603. Function qspi_init	409
Table 3-604. Function quad_spi_enable	410
Table 3-605. Function qspi_disable	410
Table 3-606. Function qspi_dma_enable	411
Table 3-607. Function qspi_dma_disable	411
Table 3-608. Function qspi_command	412
Table 3-609. Function qspi_transmit	413
Table 3-610. Function qspi_receive	413

Table 3-611. Function <code>qspi_command</code>	414
Table 3-612. Function <code>qspi_memorymapped</code>	415
Table 3-613. Function <code>qspi_abort</code>	416
Table 3-614. Function <code>qspi_command_fmc_s</code>	417
Table 3-615. Function <code>qspi_transmit_fmc_s</code>	417
Table 3-616. Function <code>qspi_data_receive_fmc_s</code>	418
Table 3-617. Function <code>qspi_command_fmc_s</code>	418
Table 3-618. Function <code>qspi_command_fmc_s</code>	420
Table 3-619. Function <code>qspi_interrupt_enable</code>	421
Table 3-620. Function <code>qspi_interrupt_disable</code>	421
Table 3-621. Function <code>qspi_flag_get</code>	422
Table 3-622. Function <code>qspi_flag_clear</code>	423
Table 3-623. RCU Registers	423
Table 3-624. RCU firmware function	424
Table 3-625. Enum <code>rcu_periph_enum</code>	426
Table 3-626. Enum <code>rcu_periph_sleep_enum</code>	428
Table 3-627. Enum <code>rcu_periph_reset_enum</code>	429
Table 3-628. Enum <code>rcu_flag_enum</code>	430
Table 3-629. Enum <code>rcu_int_flag_enum</code>	431
Table 3-630. Enum <code>rcu_int_flag_clear_enum</code>	431
Table 3-631. Enum <code>rcu_int_enum</code>	432
Table 3-632. Enum <code>rcu_osci_type_enum</code>	432
Table 3-633. Enum <code>rcu_clock_freq_enum</code>	432
Table 3-634. Enum <code>rcu_sec_enum</code>	432
Table 3-635. Enum <code>rcu_sec_flag_enum</code>	433
Table 3-636. Enum <code>rcu_unit_enum</code>	435
Table 3-637. Function <code>rcu_deinit</code>	436
Table 3-638. Function <code>rcu_periph_clock_enable</code>	436
Table 3-639. Function <code>rcu_periph_clock_disable</code>	436
Table 3-640. Function <code>rcu_periph_clock_sleep_enable</code>	437
Table 3-641. Function <code>rcu_periph_clock_sleep_disable</code>	437
Table 3-642. Function <code>rcu_periph_reset_enable</code>	438
Table 3-643. Function <code>rcu_periph_reset_disable</code>	438
Table 3-644. Function <code>rcu_bkp_reset_enable</code>	439
Table 3-645. Function <code>rcu_bkp_reset_disable</code>	439
Table 3-646. Function <code>rcu_hxtal_plli2s_enable</code>	440
Table 3-647. Function <code>rcu_hxtal_plli2s_disable</code>	440
Table 3-648. Function <code>rcu_hxtal_pll_p_enable</code>	441
Table 3-649. Function <code>rcu_hxtal_pll_p_disable</code>	441
Table 3-650. Function <code>rcu_control_unit_powerup</code>	442
Table 3-651. Function <code>rcu_control_unit_powerdown</code>	442
Table 3-652. Function <code>rcu_rfppl_cal_enable</code>	443
Table 3-653. Function <code>rcu_rfppl_cal_disable</code>	443
Table 3-654. Function <code>rcu_system_clock_source_config</code>	444

Table 3-655. Function rcu_system_clock_source_get	445
Table 3-656. Function rcu_ahb_clock_config	445
Table 3-657. Function rcu_apb1_clock_config	446
Table 3-658. Function rcu_apb2_clock_config	446
Table 3-659. Function rcu_ckout0_config	447
Table 3-660. Function rcu_ckout1_config	448
Table 3-661. Function rcu_pll_config	448
Table 3-662. Function rcu_plli2s_config	449
Table 3-663. Function rcu_pllpresel_config.....	450
Table 3-664. Function rcu_plldig_div_sys_config.....	451
Table 3-665. Function rcu_rtc_clock_config.....	451
Table 3-666. Function rcu_rtc_div_config.....	452
Table 3-667. Function rcu_i2s_clock_config	452
Table 3-668. Function rcu_pll_div_i2s_config	453
Table 3-669. Function rcu_hpdc_clock_config	453
Table 3-670. Function rcu_hpdc_audio_clock_config.....	454
Table 3-671. Function rcu_sdio_clock_config.....	455
Table 3-672. Function rcu_sdio_div_config.....	455
Table 3-673. Function rcu_usbfs_clock_config.....	456
Table 3-674. Function rcu_usbfs_div_config.....	456
Table 3-675. Function rcu_i2c0_clock_config	457
Table 3-676. Function rcu_usart0_clock_config	457
Table 3-677. Function rcu_usart2_clock_config	458
Table 3-678. Function rcu_irc16m_div_config.....	459
Table 3-679. Function rcu_sdio_div_config.....	459
Table 3-680. Function rcu_lxtal_drive_capability_config.....	460
Table 3-681. Function rcu_osci_stab_wait.....	461
Table 3-682. Function rcu_osci_on.....	461
Table 3-683. Function rcu_osci_off.....	462
Table 3-684. Function rcu_osci_bypass_mode_enable.....	462
Table 3-685. Function rcu_osci_bypass_mode_disable.....	463
Table 3-686. Function rcu_hxtal_clock_monitor_enable.....	463
Table 3-687. Function rcu_hxtal_clock_monitor_disable.....	464
Table 3-688. Function rcu_hxtal_clock_monitor_enable.....	464
Table 3-689. Function rcu_hxtal_clock_monitor_disable.....	464
Table 3-690. Function rcu_IRC16M_adjust_value_set	465
Table 3-691. Function rcu_spread_spectrum_config	465
Table 3-692. Function rcu_spread_spectrum_enable.....	466
Table 3-693. Function rcu_spread_spectrum_disable.....	467
Table 3-694. Function rcu_voltage_key_unlock	467
Table 3-695. Function rcu_deepsleep_voltage_set.....	468
Table 3-696. Function rcu_clock_freq_get.....	468
Table 3-697. Function rcu_security_enable	469
Table 3-698. Function rcu_security_disable	469

Table 3-699. Function rcu_privilege_enable	470
Table 3-700. Function rcu_privilege_disable	470
Table 3-701. Function rcu_flag_get.....	471
Table 3-702. Function rcu_all_reset_flag_clear	471
Table 3-703. Function rcu_interrupt_flag_get.....	472
Table 3-704. Function rcu_interrupt_flag_clear	472
Table 3-705. Function rcu_security_flag_get.....	473
Table 3-706. Function rcu_interrupt_enable	474
Table 3-707. Function rcu_interrupt_disable	475
Table 3-708. RTC Registers	475
Table 3-709. RTC firmware function.....	476
Table 3-710. Structure rtc_parameter_struct	478
Table 3-711. Structure rtc_alarm_struct	478
Table 3-712. Structure rtc_timestamp_struct	478
Table 3-713. Structure rtc_tamper_struct.....	478
Table 3-714. Function rtc_deinit.....	479
Table 3-715. Function rtc_init	479
Table 3-716. Function rtc_init_mode_enter	480
Table 3-717. Function rtc_init_mode_exit	481
Table 3-718. Function rtc_register_sync_wait.....	481
Table 3-719. Function rtc_current_time_get	482
Table 3-720. Function rtc_subsecond_get.....	482
Table 3-721. Function rtc_alarm_config.....	483
Table 3-722. Function rtc_alarm_subsecond_config.....	483
Table 3-723. Function rtc_alarm_get	485
Table 3-724. Function rtc_alarm_subsecond_get	485
Table 3-725. Function rtc_alarm_enable	486
Table 3-726. Function rtc_alarm_disable	486
Table 3-727. Function rtc_timestamp_enable.....	487
Table 3-728. Function rtc_timestamp_disable	487
Table 3-729. Function rtc_timestamp_get.....	488
Table 3-730. Function rtc_timestamp_subsecond_get.....	489
Table 3-731. Function rtc_tamper_enable	489
Table 3-732. Function rtc_tamper_disable.....	490
Table 3-733. Function rtc_software_bkp_reset	490
Table 3-734. Function rtc_tamper_without_bkp_seset.....	491
Table 3-735. Function rtc_interrupt_enable	491
Table 3-736. Function rtc_interrupt_disable	492
Table 3-737. Function rtc_flag_get.....	492
Table 3-738. Function rtc_flag_clear.....	493
Table 3-739. Function rtc_nsec_interrupt_flag_get	494
Table 3-740. Function rtc_nsec_interrupt_flag_clear	495
Table 3-741. Function rtc_sec_interrupt_flag_get.....	496
Table 3-742. Function rtc_nsec_interrupt_flag_clear	496

Table 3-743. Function rtc_output_pad_select	497
Table 3-744. Function rtc_alter_output_config	498
Table 3-745. Function rtc_calibration_config	498
Table 3-746. Function rtc_hour_adjust	499
Table 3-747. Function rtc_second_adjust	500
Table 3-748. Function rtc_bypass_shadow_enable	500
Table 3-749. Function rtc_bypass_shadow_disable	501
Table 3-750. Function rtc_refclock_detection_enable	501
Table 3-751. Function rtc_refclock_detection_disable	502
Table 3-752. Function rtc_wakeup_enable	502
Table 3-753. Function rtc_wakeup_disable	503
Table 3-754. Function rtc_wakeup_clock_set	503
Table 3-755. Function rtc_wakeup_timer_set	504
Table 3-756. Function rtc_wakeup_timer_get	504
Table 3-757. rtc_smooth_calibration_config	505
Table 3-758. rtc_coarse_calibration_enable	506
Table 3-759. rtc_coarse_calibration_disable	506
Table 3-760. rtc_coarse_calibration_config	507
Table 3-761. rtc_pri_pro_enable	507
Table 3-762. rtc_pri_pro_enable	508
Table 3-763. rtc_sec_pro_enable	509
Table 3-764. rtc_sec_pro_disable	510
Table 3-765. rtc_bkp_zone_a_sec_pro_set	511
Table 3-766. rtc_bkp_zone_b_sec_pro_set	512
Table 3-767. rtc_bkp_zone_b_sec_pro_check	512
Table 3-768. SDIO Registers	513
Table 3-769. SDIO firmware function	514
Table 3-770. Function sdio_deinit	515
Table 3-771. Function sdio_clock_config	515
Table 3-772. Function sdio_hardware_clock_enable	516
Table 3-773. Function sdio_hardware_clock_disable	517
Table 3-774. Function sdio_bus_mode_set	517
Table 3-775. Function sdio_power_state_set	518
Table 3-776. Function sdio_power_state_get	519
Table 3-777. Function sdio_clock_enable	519
Table 3-778. Function sdio_clock_disable	520
Table 3-779. Function sdio_command_response_config	520
Table 3-780. Function sdio_wait_type_set	521
Table 3-781. Function sdio_csm_enable	521
Table 3-782. Function sdio_csm_disable	522
Table 3-783. Function sdio_command_index_get	522
Table 3-784. Function sdio_response_get	523
Table 3-785. Function sdio_data_config	523
Table 3-786. Function sdio_data_transfer_config	525

Table 3-787. Function <code>sdio_dsm_enable</code>	525
Table 3-788. Function <code>sdio_dsm_disable</code>	526
Table 3-789. Function <code>sdio_data_write</code>	526
Table 3-790. Function <code>sdio_data_read</code>	527
Table 3-791. Function <code>sdio_data_counter_get</code>	527
Table 3-792. Function <code>sdio_data_counter_get</code>	528
Table 3-793. Function <code>sdio_dma_enable</code>	528
Table 3-794. Function <code>sdio_dma_disable</code>	529
Table 3-795. Function <code>sdio_flag_get</code>	529
Table 3-796. Function <code>sdio_flag_clear</code>	531
Table 3-797. Function <code>sdio_interrupt_enable</code>	532
Table 3-798. Function <code>sdio_interrupt_disable</code>	533
Table 3-799. Function <code>sdio_interrupt_flag_get</code>	534
Table 3-800. Function <code>sdio_interrupt_flag_clear</code>	535
Table 3-801. Function <code>sdio_readwait_enable</code>	536
Table 3-802. Function <code>sdio_readwait_disable</code>	537
Table 3-803. Function <code>sdio_stop_readwait_enable</code>	537
Table 3-804. Function <code>sdio_stop_readwait_disable</code>	538
Table 3-805. Function <code>sdio_readwait_type_set</code>	538
Table 3-806. Function <code>sdio_operation_enable</code>	539
Table 3-807. Function <code>sdio_operation_disable</code>	539
Table 3-808. Function <code>sdio_suspend_enable</code>	540
Table 3-809. Function <code>sdio_suspend_disable</code>	540
Table 3-810. Function <code>sdio_ceata_command_enable</code>	541
Table 3-811. Function <code>sdio_ceata_command_disable</code>	541
Table 3-812. Function <code>sdio_ceata_interrupt_enable</code>	542
Table 3-813. Function <code>sdio_ceata_interrupt_disable</code>	542
Table 3-814. Function <code>sdio_ceata_command_completion_enable</code>	543
Table 3-815. Function <code>sdio_ceata_command_completion_disable</code>	543
Table 3-816. SPI/I2S Registers.....	544
Table 3-817. SPI/I2S firmware function.....	544
Table 3-818. Structure <code>spi_parameter_struct</code>	545
Table 3-819. Function <code>spi_i2s_deinit</code>	546
Table 3-820. Function <code>spi_struct_para_init</code>	546
Table 3-821. Function <code>spi_init</code>	547
Table 3-822. Function <code>spi_enable</code>	548
Table 3-823. Function <code>spi_disable</code>	548
Table 3-824. Function <code>i2s_init</code>	549
Table 3-825. Function <code>i2s_psc_config</code>	550
Table 3-826. Function <code>i2s1_ckin_psc_config</code>	551
Table 3-827. Function <code>i2s_enable</code>	552
Table 3-828. Function <code>i2s_disable</code>	553
Table 3-829. Function <code>spi_nss_output_enable</code>	553
Table 3-830. Function <code>spi_nss_output_disable</code>	554

Table 3-831. Function spi_nss_internal_high	554
Table 3-832. Function spi_nss_internal_low	555
Table 3-833. Function spi_dma_enable	555
Table 3-834. Function spi_dma_disable	556
Table 3-835. Function spi_i2s_data_frame_format_config	557
Table 3-836. Function spi_i2s_data_transmit	557
Table 3-837. Function spi_i2s_data_receive	558
Table 3-838. Function spi_bidirectional_transfer_config	558
Table 3-839. Function i2s_init	559
Table 3-840. Function spi_i2s_format_error_clear	560
Table 3-841. Function spi_crc_polynomial_set	561
Table 3-842. Function spi_crc_polynomial_get	562
Table 3-843. Function spi_crc_on	562
Table 3-844. Function spi_crc_off	563
Table 3-845. Function spi_crc_next	563
Table 3-846. Function spi_crc_get	564
Table 3-847. Function spi_crc_error_clear	564
Table 3-848. Function spi_ti_mode_enable	565
Table 3-849. Function spi_ti_mode_disable	565
Table 3-850. Function spi_quad_enable	566
Table 3-851. Function spi_quad_disable	566
Table 3-852. Function spi_quad_write_enable	567
Table 3-853. Function spi_quad_read_enable	567
Table 3-854. Function spi_quad_io23_output_enable	568
Table 3-855. Function spi_quad_io23_output_disable	568
Table 3-856. Function spi_i2s_flag_get	569
Table 3-857. Function spi_i2s_interrupt_enable	570
Table 3-858. Function spi_i2s_interrupt_disable	570
Table 3-859. Function spi_i2s_interrupt_flag_get	571
Table 3-860. SQPI Registers	572
Table 3-861. SQPI firmware function	572
Table 3-862. sqpi_parameter_struct	573
Table 3-863. Function sqpi_deinit	573
Table 3-864. Function sqpi_struct_para_init	574
Table 3-865. Function sqpi_init	574
Table 3-866. Function sqpi_read_id_command	575
Table 3-867. Function sqpi_special_command	575
Table 3-868. Function sqpi_read_command_config	576
Table 3-869. Function sqpi_write_command_config	577
Table 3-870. Function sqpi_low_id_receive	577
Table 3-871. Function sqpi_low_id_receive	578
Table 3-872. SYSCFG registers	578
Table 3-873. SYSCFG firmware function	579
Table 3-874. Function syscfg_deinit	580

Table 3-875. Function syscfg_exti_line_config	580
Table 3-876. Function compensation_pwdn_mode_enable	581
Table 3-877. Function compensation_pwdn_mode_disable	581
Table 3-878. Function syscfg_clock_access_security_config	582
Table 3-879. Function classb_access_security_config	582
Table 3-880. Function sram1_access_security_config	583
Table 3-881. Function fpu_access_security_config	584
Table 3-882. Function vtor_ns_write_disable	584
Table 3-883. Function mpu_ns_write_disable	585
Table 3-884. Function vtors_aircr_write_disable	585
Table 3-885. Function vtors_aircr_write_disable	586
Table 3-886. Function sau_write_disable	586
Table 3-887. Function syscfg_lock_config	587
Table 3-888. Function gssacmd_write_data	587
Table 3-889. Function sram1_erase	588
Table 3-890. Function sram1_unlock	588
Table 3-891. Function sram1_lock	589
Table 3-892. Function sram1_write_protect_0_31	589
Table 3-893. Function sram1_write_protect_32_63	590
Table 3-894. Function compensation_ready_flag_get	590
Table 3-895. Function sram1_bsy_flag_get	591
Table 3-896. Function fpu_interrupt_enable	591
Table 3-897. Function fpu_interrupt_disable	592
Table 3-898. TIMERx Registers	593
Table 3-899. TIMERx firmware function	593
Table 3-900. Structure timer_parameter_struct	596
Table 3-901. Structure timer_break_parameter_struct	596
Table 3-902. Structure timer_oc_parameter_struct	596
Table 3-903. Structure timer_ic_parameter_struct	597
Table 3-904. Function timer_deinit	597
Table 3-905. Function timer_struct_para_init	598
Table 3-906. Function timer_init	598
Table 3-907. Function timer_enable	599
Table 3-908. Function timer_disable	600
Table 3-909. Function timer_auto_reload_shadow_enable	600
Table 3-910. Function timer_auto_reload_shadow_disable	601
Table 3-911. Function timer_update_event_enable	601
Table 3-912. Function timer_update_event_disable	602
Table 3-913. Function timer_counter_alignment	602
Table 3-914. Function timer_counter_up_direction	603
Table 3-915. timer_counter_down_direction	603
Table 3-916. Function timer_prescaler_config	604
Table 3-917. Function timer_repetition_value_config	605
Table 3-918. Function timer_autoreload_value_config	605

Table 3-919. Function timer_counter_value_config.....	606
Table 3-920. Function timer_counter_read	606
Table 3-921. Function timer_prescaler_read	607
Table 3-922. Function timer_single_pulse_mode_config.....	607
Table 3-923. Function timer_update_source_config.....	608
Table 3-924. Function timer_dma_enable	609
Table 3-925. Function timer_dma_disable	609
Table 3-926. Function timer_channel_dma_request_source_select.....	610
Table 3-927. Function timer_dma_transfer_config	611
Table 3-928. Function timer_event_software_generate.....	613
Table 3-929. Function timer_break_struct_para_init	613
Table 3-930. Function timer_break_config.....	614
Table 3-931. Function timer_break_enable	615
Table 3-932. Function timer_break_disable	615
Table 3-933. Function timer_automatic_output_enable	616
Table 3-934. Function timer_automatic_output_disable	617
Table 3-935. Function timer_primary_output_config.....	617
Table 3-936. Function timer_channel_control_shadow_config.....	618
Table 3-937. Function timer_channel_control_shadow_update_config	618
Table 3-938. Function timer_channel_output_struct_para_init	619
Table 3-939. Function timer_channel_output_config	620
Table 3-940. Function timer_channel_output_mode_config.....	621
Table 3-941. Function timer_channel_output_pulse_value_config.....	622
Table 3-942. Function timer_channel_output_shadow_config	622
Table 3-943. Function timer_channel_output_fast_config.....	623
Table 3-944. Function timer_channel_output_clear_config.....	624
Table 3-945. Function timer_channel_output_polarity_config	625
Table 3-946. Function timer_channel_complementary_output_polarity_config.....	626
Table 3-947. Function timer_channel_output_state_config.....	626
Table 3-948. Function timer_channel_complementary_output_state_config	627
Table 3-949. Function timer_channel_input_struct_para_init.....	628
Table 3-950. Function timer_input_capture_config	629
Table 3-951. Function timer_channel_input_capture_prescaler_config.....	629
Table 3-952. Function timer_channel_capture_value_register_read	630
Table 3-953. Function timer_input_pwm_capture_config	631
Table 3-954. Function timer_hall_mode_config	632
Table 3-955. Function timer_input_trigger_source_select.....	632
Table 3-956. Function timer_master_output_trigger_source_select	633
Table 3-957. Function timer_slave_mode_select	635
Table 3-958. Function timer_master_slave_mode_config.....	635
Table 3-959. Function timer_external_trigger_config.....	636
Table 3-960. Function timer_quadrature_decoder_mode_config.....	637
Table 3-961. Function timer_internal_clock_config.....	638
Table 3-962. Function timer_internal_trigger_as_external_clock_config.....	639

Table 3-963. Function timer_external_trigger_as_external_clock_config	639
Table 3-964. Function timer_external_clock_mode0_config	640
Table 3-965. Function timer_external_clock_mode1_config	641
Table 3-966. Function timer_external_clock_mode1_disable	642
Table 3-967. Function timer_write_chxval_register_config	643
Table 3-968. Function timer_output_value_selection_config	643
Table 3-969. Function timer_flag_get	644
Table 3-970. Function timer_flag_clear	645
Table 3-971. Function timer_interrupt_enable	646
Table 3-972. Function timer_interrupt_disable	647
Table 3-973. Function timer_interrupt_flag_get	647
Table 3-974. Function timer_interrupt_flag_clear	648
Table 3-975 TRNG Registers	649
Table 3-976. TRNG firmware function	649
Table 3-977. Enum trng_flag_enum	650
Table 3-978. Enum trng_int_flag_enum	650
Table 3-979. Function trng_deinit	650
Table 3-980. Function trng_enable	650
Table 3-981 Function trng_disable	651
Table 3-982 Function trng_get_true_random_data	651
Table 3-983 trng_interrupt_enable	652
Table 3-984 trng_interrupt_disable	652
Table 3-985 trng_flag_get	653
Table 3-986 trng_interrupt_flag_get	653
Table 3-987 trng_interrupt_flag_clear	654
Table 3-988. TSI Registers	655
Table 3-989. TSI firmware function	655
Table 3-990. Function tsi_deinit	656
Table 3-991. Function tsi_init	657
Table 3-992. Function tsi_enable	658
Table 3-993. Function tsi_disable	658
Table 3-994. Function tsi_sample_pin_enable	659
Table 3-995. Function tsi_sample_pin_disable	659
Table 3-996. Function tsi_channel_pin_enable	660
Table 3-997. Function tsi_channel_pin_disable	660
Table 3-998. Function tsi_software_mode_config	661
Table 3-999. Function tsi_software_start	661
Table 3-1000. Function tsi_software_stop	662
Table 3-1001. Function tsi_hardware_mode_config	662
Table 3-1002. Function tsi_pin_mode_config	663
Table 3-1003. Function tsi_extend_charge_config	664
Table 3-1004. Function tsi_plus_config	664
Table 3-1005. Function tsi_max_number_config	666
Table 3-1006. Function tsi_hysteresis_on	666

Table 3-1007. Function tsi_hysteresis_off	667
Table 3-1008. Function tsi_analog_on	667
Table 3-1009. Function tsi_analog_off	668
Table 3-1010. Function tsi_group_enable	668
Table 3-1011. Function tsi_group_disable	669
Table 3-1012. Function tsi_group_status_get	669
Table 3-1013. Function tsi_group0_cycle_get	670
Table 3-1014. Function tsi_group1_cycle_get	670
Table 3-1015. Function tsi_group2_cycle_get	671
Table 3-1016. Function tsi_flag_clear	671
Table 3-1017. Function tsi_flag_get	672
Table 3-1018. Function tsi_interrupt_enable	672
Table 3-1019. Function tsi_interrupt_disable	673
Table 3-1020. Function tsi_interrupt_flag_clear	674
Table 3-1021. Function tsi_interrupt_flag_get	674
Table 3-1022. TZPCU Registers	675
Table 3-1023. TZPCU firmware function	676
Table 3-1024. tzpcu_mem	676
Table 3-1025. tzpcu_non_secure_mark_region	676
Table 3-1026. tzpcu_non_secure_mark_struct	677
Table 3-1027. Function tzpcu_tzspc_peripheral_attributes_config	677
Table 3-1028. Function tzpcu_tzspc_peripheral_attributes_get	679
Table 3-1029. Function tzpcu_non_secure_mark_struct_para_init	681
Table 3-1030. Function tzpcu_tzspc_emnsm_config	681
Table 3-1031. Function tzpcu_tzspc_items_lock	682
Table 3-1032. Function tzpcu_tzspc_dbg_config	683
Table 3-1033. Function tzpcu_tzmpc_lock	683
Table 3-1034. Function tzpcu_tzmpc_security_state_config	684
Table 3-1035. Function tzpcu_tzmpc_secure_access_config	684
Table 3-1036. Function tzpcu_tzmpc_block_secure_access_mode_config	685
Table 3-1037. Function tzpcu_tzmpc_union_block_lock	686
Table 3-1038. Function tzpcu_tziac_interrupt_enable	687
Table 3-1039. Function tzpcu_tziac_interrupt_disable	689
Table 3-1040. Function tzpcu_tziac_flag_get	692
Table 3-1041. Function tzpcu_tziac_flag_clear	695
Table 3-1042. USART Registers	698
Table 3-1043. USART firmware function	699
Table 3-1044. Enum usart_flag_enum	701
Table 3-1045. Enum usart_interrupt_flag_enum	702
Table 3-1046. Enum usart_interrupt_enum	702
Table 3-1047. Enum usart_invert_enum	703
Table 3-1048. Function usart_deinit	703
Table 3-1049. Function usart_baudrate_set	703
Table 3-1050. Function usart_parity_config	704

Table 3-1051. Function usart_word_length_set	705
Table 3-1052. Function usart_stop_bit_set	705
Table 3-1053. Function usart_enable	706
Table 3-1054. Function usart_disable	706
Table 3-1055. Function usart_transmit_config	707
Table 3-1056. Function usart_receive_config	707
Table 3-1057. Function usart_data_first_config	708
Table 3-1058. Function usart_invert_config	709
Table 3-1059. Function usart_overrun_enable	710
Table 3-1060. Function usart_overrun_disable	710
Table 3-1061. Function usart_oversample_config	711
Table 3-1062. Function usart_sample_bit_config	711
Table 3-1063. Function usart_receiver_timeout_enable	712
Table 3-1064. Function usart_receiver_timeout_disable	712
Table 3-1065. Function usart_receiver_timeout_threshold_config	713
Table 3-1066. Function usart_data_transmit	713
Table 3-1067. Function usart_data_receive	714
Table 3-1068. Function usart_address_config	715
Table 3-1069. Function usart_address_detection_mode_config	715
Table 3-1070. Function usart_mute_mode_enable	716
Table 3-1071. Function usart_mute_mode_disable	716
Table 3-1072. Function usart_mute_mode_wakeup_config	717
Table 3-1073. Function usart_lin_mode_enable	717
Table 3-1074. Function usart_lin_mode_disable	718
Table 3-1075. Function usart_lin_break_detection_length_config	718
Table 3-1076. Function usart_halfduplex_enable	719
Table 3-1077. Function usart_halfduplex_disable	719
Table 3-1078. Function usart_clock_enable	720
Table 3-1079. Function usart_clock_disable	720
Table 3-1080. Function usart_synchronous_clock_config	721
Table 3-1081. Function usart_guard_time_config	722
Table 3-1082. Function usart_smartcard_mode_enable	722
Table 3-1083. Function usart_smartcard_mode_disable	723
Table 3-1084. Function usart_smartcard_mode_nack_enable	723
Table 3-1085. Function usart_smartcard_mode_nack_disable	724
Table 3-1086. Function usart_smartcard_mode_early_nack_enable	724
Table 3-1087. Function usart_smartcard_mode_early_nack_disable	725
Table 3-1088. Function usart_smartcard_autoretry_config	725
Table 3-1089. Function usart_block_length_config	726
Table 3-1090. Function usart_irda_mode_enable	726
Table 3-1091. Function usart_irda_mode_disable	727
Table 3-1092. Function usart_prescaler_config	727
Table 3-1093. Function usart_irda_lowpower_config	728
Table 3-1094. Function usart_hardware_flow_rts_config	729

Table 3-1095. Function usart_hardware_flow_cts_config.....	729
Table 3-1096. Function usart_hardware_flow_coherence_config.....	730
Table 3-1097. Function usart_rs45_driver_enable	731
Table 3-1098. Function usart_rs45_driver_disable	731
Table 3-1099. Function usart_driver_asserttime_config	732
Table 3-1100. Function usart_driver_deasserttime_config	732
Table 3-1101. Function usart_depolarity_config	733
Table 3-1102. Function usart_dma_receive_config.....	733
Table 3-1103. Function usart_dma_transmit_config.....	734
Table 3-1104. Function usart_reception_error_dma_disable.....	735
Table 3-1105. Function usart_reception_error_dma_enable.....	735
Table 3-1106. Function usart_wakeup_enable.....	736
Table 3-1107. Function usart_wakeup_disable.....	736
Table 3-1108. Function usart_wakeup_mode_config.....	737
Table 3-1109. Function usart_receive_fifo_enable.....	737
Table 3-1110. Function usart_receive_fifo_disable	738
Table 3-1111. Function usart_receive_fifo_counter_number	738
Table 3-1112. Function usart_command_enable	739
Table 3-1113. Function usart_flag_get.....	740
Table 3-1114. Function usart_flag_clear.....	741
Table 3-1115. Function usart_interrupt_enable	742
Table 3-1116. Function usart_interrupt_disable	743
Table 3-1117. Function usart_interrupt_flag_get.....	743
Table 3-1118. Function usart_interrupt_flag_clear.....	745
Table 3-1119. WWDGT Registers	746
Table 3-1120. WWDGT firmware function.....	746
Table 3-1121. Function wwdgt_deinit	747
Table 3-1122. Function wwdgt_enable.....	747
Table 3-1123. Function wwdgt_counter_update.....	748
Table 3-1124. Function wwdgt_config	748
Table 3-1125. Function wwdgt_interrupt_enable.....	749
Table 3-1126. Function wwdgt_flag_get	749
Table 3-1127. Function wwdgt_flag_clear	750
Table 4-1. Revision history	751

1. Introduction

This manual introduces firmware library of GD32W51x 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 GD32W51x 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
CAU	Cryptographic acceleration unit
CRC	CRC calculation unit
DBG	Debug
DCI	Digital camera interface
DMA	Direct memory access controller
EFUSE	Electronic fuse
EXTI	Interrupt/event controller
FMC	Flash memory controller
FWDGT	Free watchdog timer
GPIO/AFIO	General-purpose and alternate-function I/Os
HAU	Hash acceleration unit
HPDF	High-performance digital filter
I2C	Inter-integrated circuit interface
ICACHE	Instruction cache

Peripherals	Descriptions
MISC	Nested Vectored Interrupt Controller
PKCAU	Public key cryptographic acceleration unit
PMU	Power management unit
QSPI	Quad-SPI interface
RCU	Reset and clock unit
RTC	Real-time Clock
SDIO	Secure digital input/output interface
SPI/I2S	Serial peripheral interface/Inter-IC sound
SQPI	Serial/Quad Parallel Interface
TIMER	TIMER
TRNG	True random number generator
TSI	Touch sensing interface
TZPCU	TrustZone protection controller union
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:

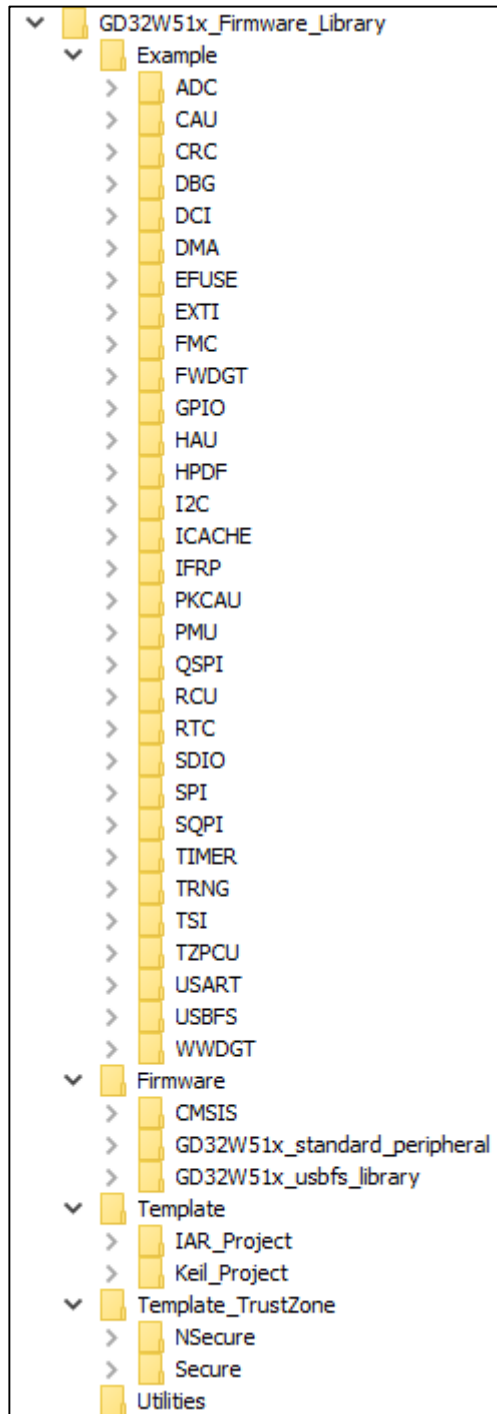
- The peripherals are shortened in XXX format, such as: ADC. More shorten information of peripherals refer to [Peripherals](#);
- The name of sourcefile and header file are started with “gd32w51x_”, such as: gd32w51x_adc.h;
- The constants used only in one file should be defined in the used file; the 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

GD32W51x_Firmware_Library, the file structure is shown as below:

Figure 2-1. File structure of firmware library of GD32W51x



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;
- gd32w51x_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);
- gd32w51x_it.c: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- gd32w51x_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.

The GD32W51x series MCU based on Arm® Cortex™-M33 core and supports Arm® Trustzone® technology. Some peripherals include trustzone enabling examples. In these examples, the following files will be added:

- partition_gd32w51x.h: the header file configures SAU/IDAU and secure/non-secure interrupts attributes;
- Project_S.sct: the keil scatter-loading file provides detailed information about the grouping and placement of each area and part of the image in the security project;
- Project_NS.sct: the keil scatter-loading file provides detailed information about the grouping and placement of each area and part of the image in the non-security project;
- gd32w51x_flash_s.icf: the IAR linker configuration file provides to link and locate an application in memory according requirements in the security project;
- gd32w51x_flash_ns.icf: the IAR linker configuration file provides to link and locate an application in memory according requirements in the non-security project;

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 M33 kernel support files, the startup file based on the Cortex M33 kernel processor, the global header file of GD32W51x and system configuration file;
- GD32W51x_standard_peripheral subfolder:
 - Include subfolder includes all the header files of firmware library, users need not modify this folder;
 - Source subfolder includes all the source files of firmware library, users need not modify this folder;
- GD32W51x_usbd_library subfolder includes all the related files about USB D peripheral:
 - Include subfolder includes the header files of USB D peripheral, users need not

- modify this folder;
- Source subfolder includes the source files of USB D peripheral, users need not modify this folder;
- GD32W51x_usbfs_library subfolder includes all the related files about USBFS peripheral:
 - Include subfolder includes the header files of USBFS peripheral, users need not modify this folder;
 - Source subfolder includes the source files of 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.

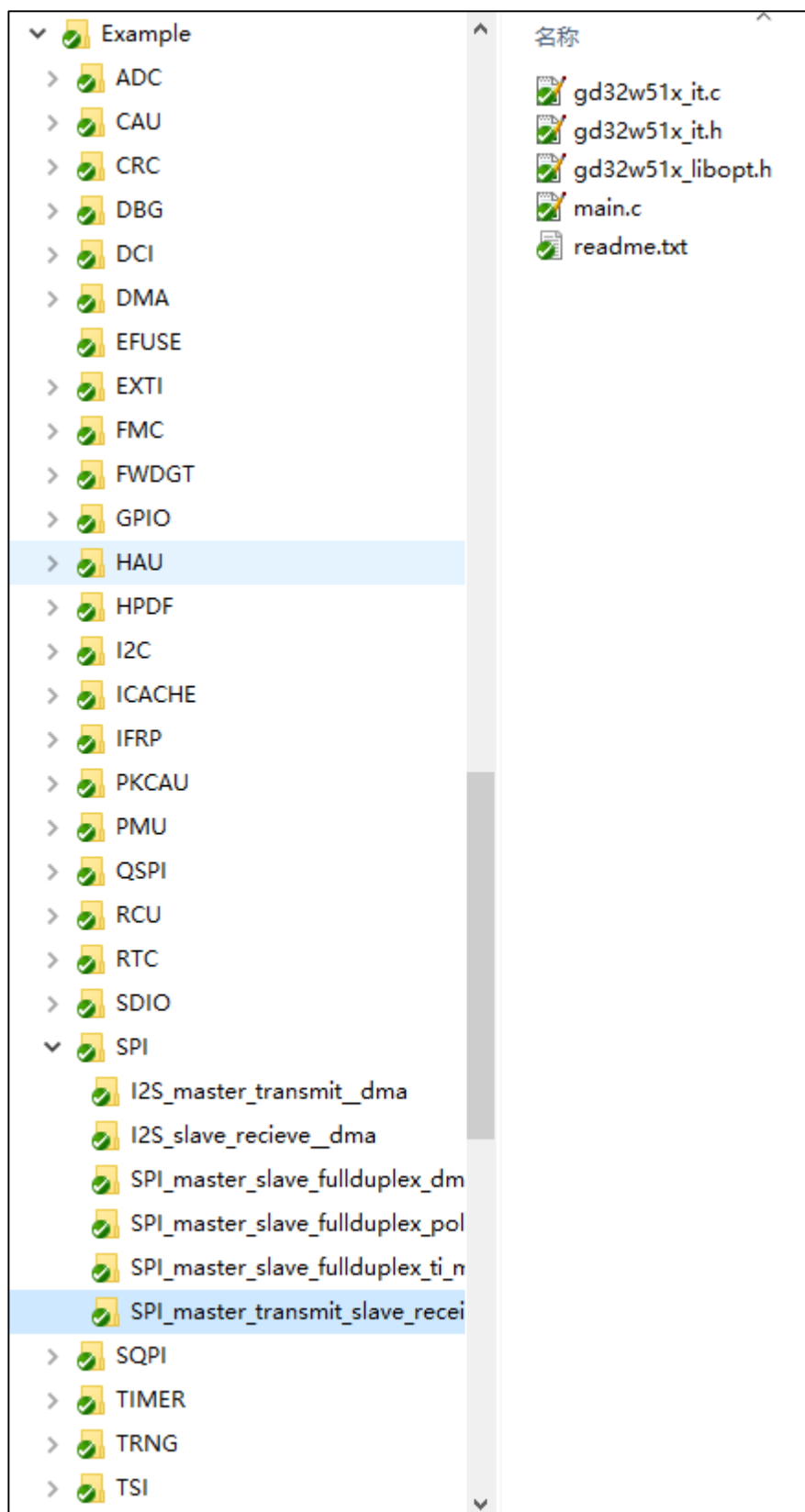
2.1.3. Template Folder

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, and Keil_project is run in Keil5). User can use the project template to compile the firmware 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:

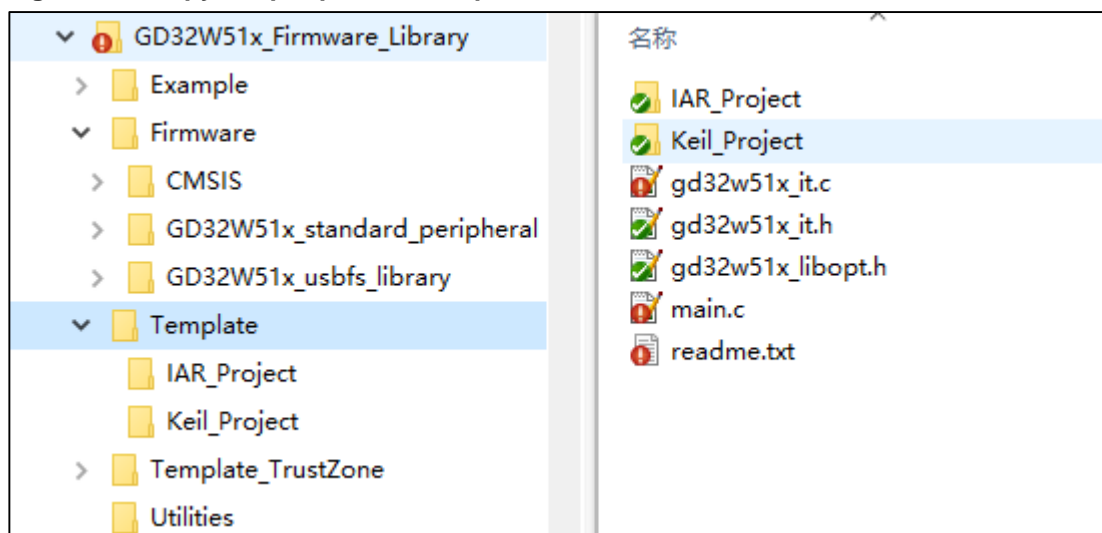
Figure 2-2. Select peripheral example files



Copy files

Open “Template” folder, keep the folders of ” IAR_project” and ” Keil_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:

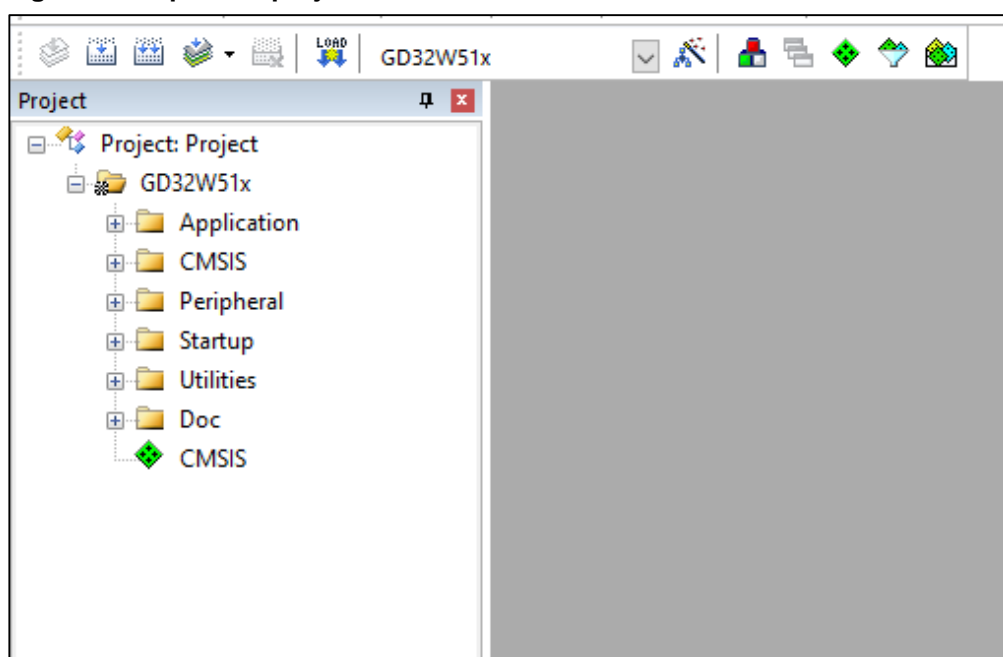
Figure 2-3. Copy the peripheral example files



Open a project

GD provides project in Keil and IAR, users can open project in different IDEs according to their need, such as "Keil_project", open \Template\Keil_project\Project.uvprojx, shown as below:

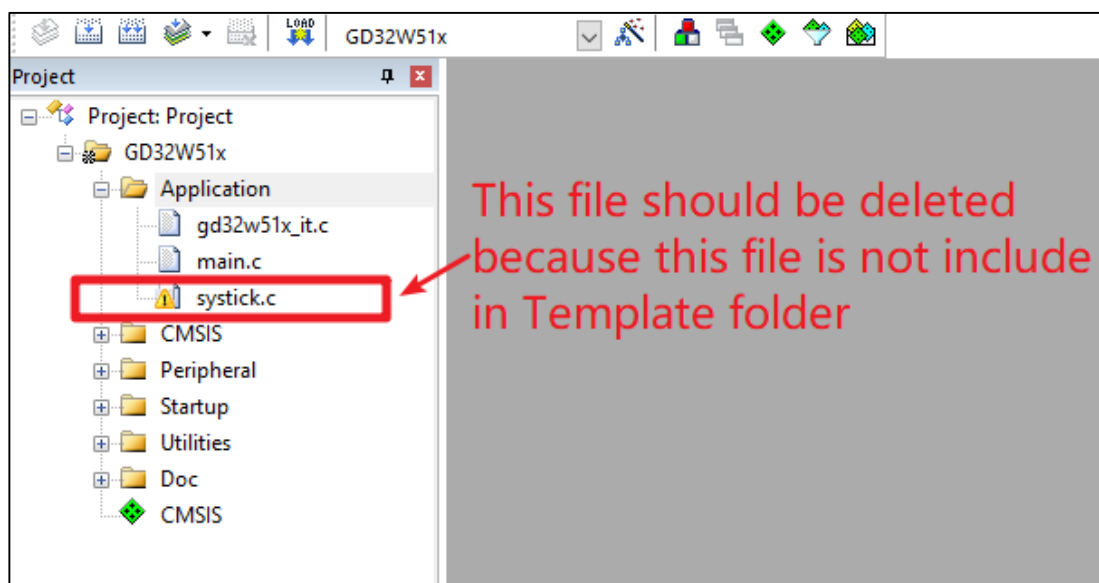
Figure 2-4. Open the project file



Because different module and different functions adopt different files, users should add or

delete the files in project according to the copied files, shown as below:

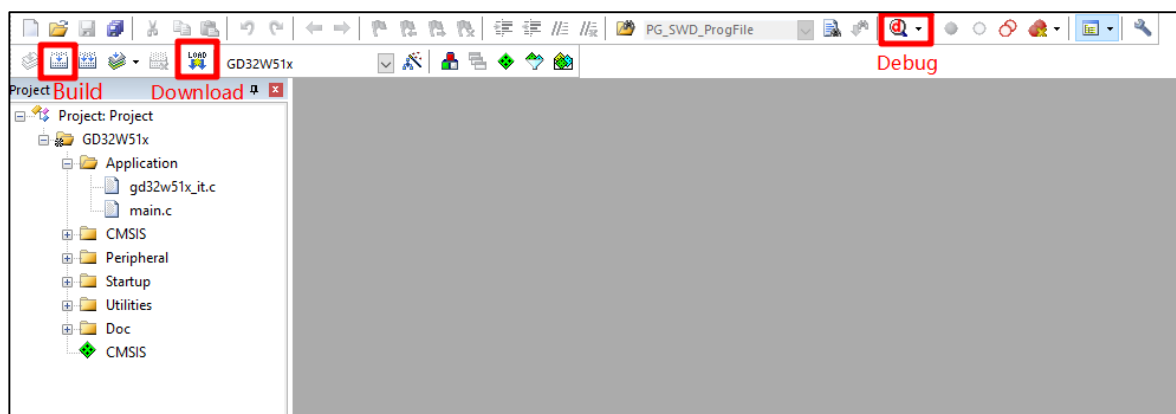
Figure 2-5. Configure project files



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 2-6. Compile-debug-download



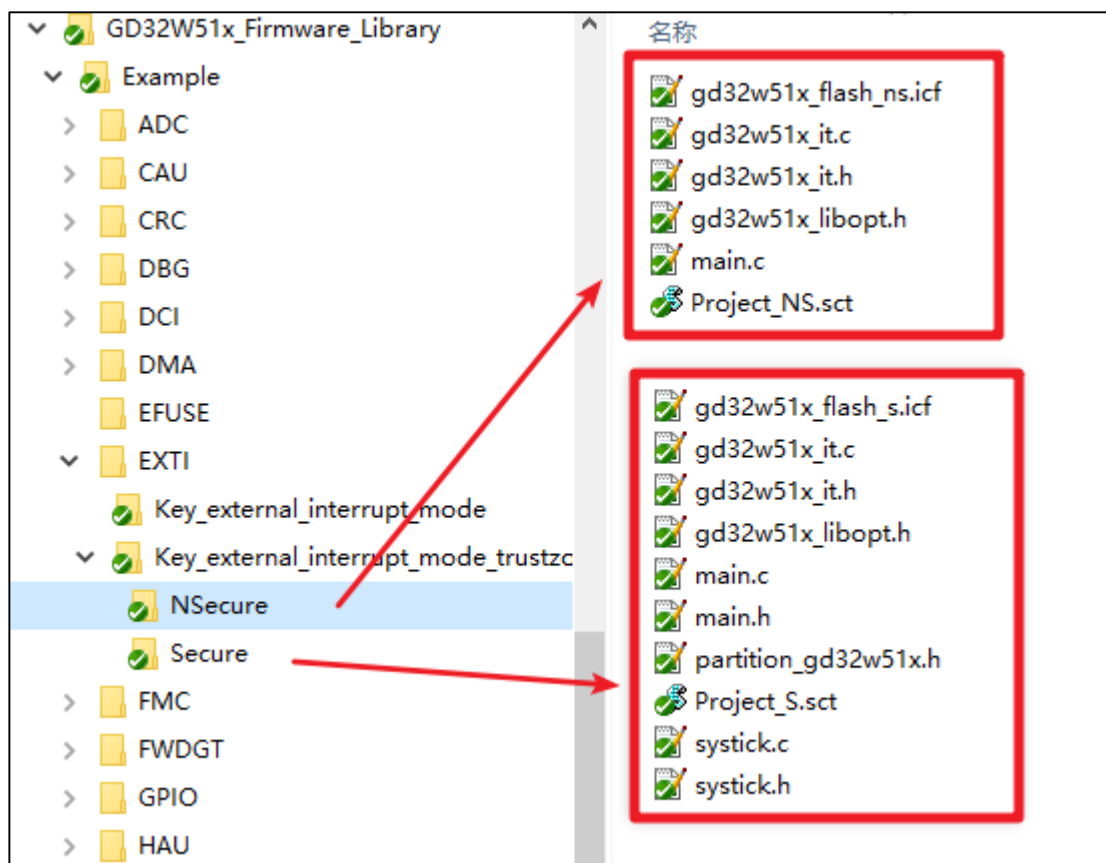
2.1.4. Template_TrustZone Folder

Template_TrustZone folder includes two subdirectories, Secure and Nsecure. A simple demo of how to switch between secure application and non-secure application, (IAR_project is run in IAR, and Keil_project is run in Keil5). User can use the project template to compile the firmware examples, the steps are shown as below:

Select files

Open “Examples” folder, select the module to be tested, such as EXTI, open “EXTI” folder, select an trustzone example of EXTI, such as “Key_external_interrupt_mode_trustzone”, shown as below:

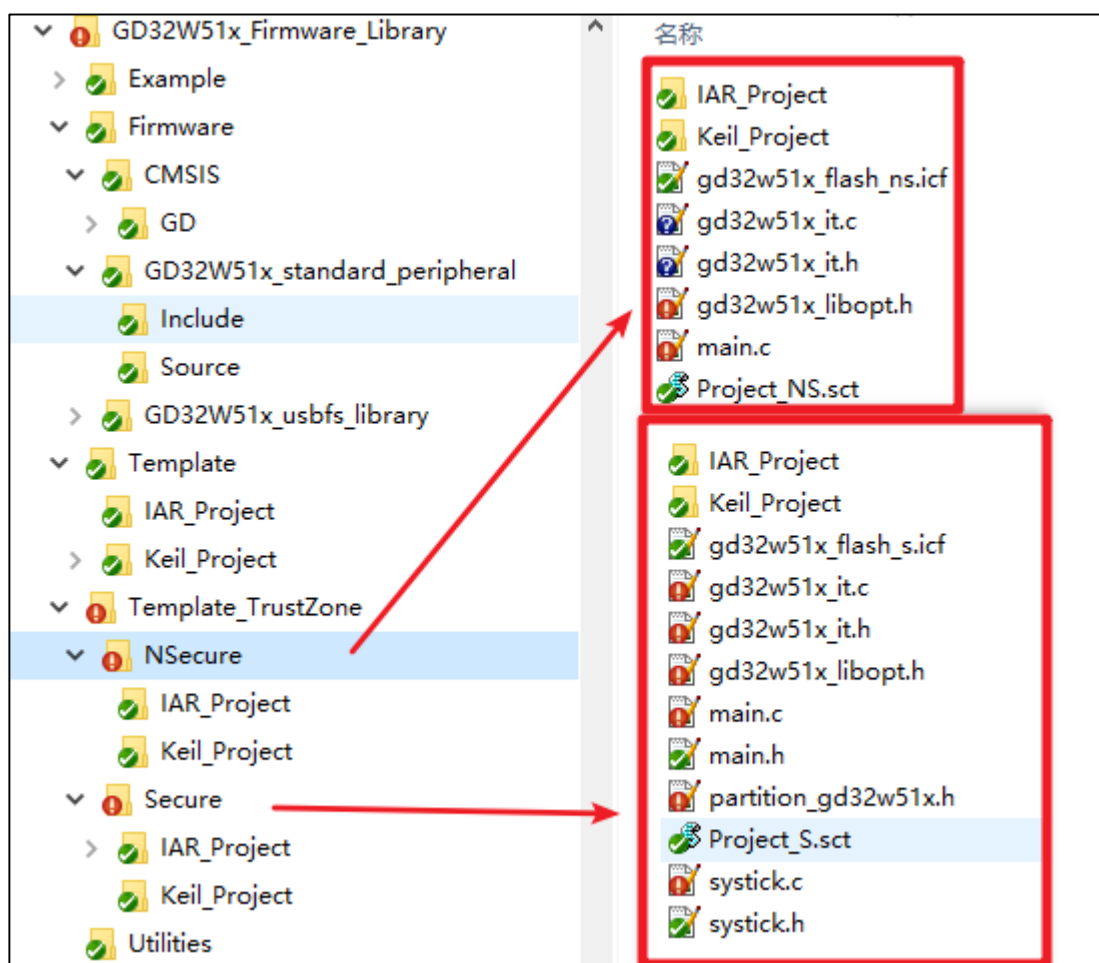
Figure 2-7. Select peripheral example files



Copy files

Open “Template_TrustZone” folder, in secure and nsecure subdirectories, keep the folders of “IAR_project” and “Keil_project”, and delete the other files, then copy all the files in “Key_external_interrupt_mode_trustzone” folder to the “Template_Trustzone” subfolder secure and nsecure separately, shown as below:

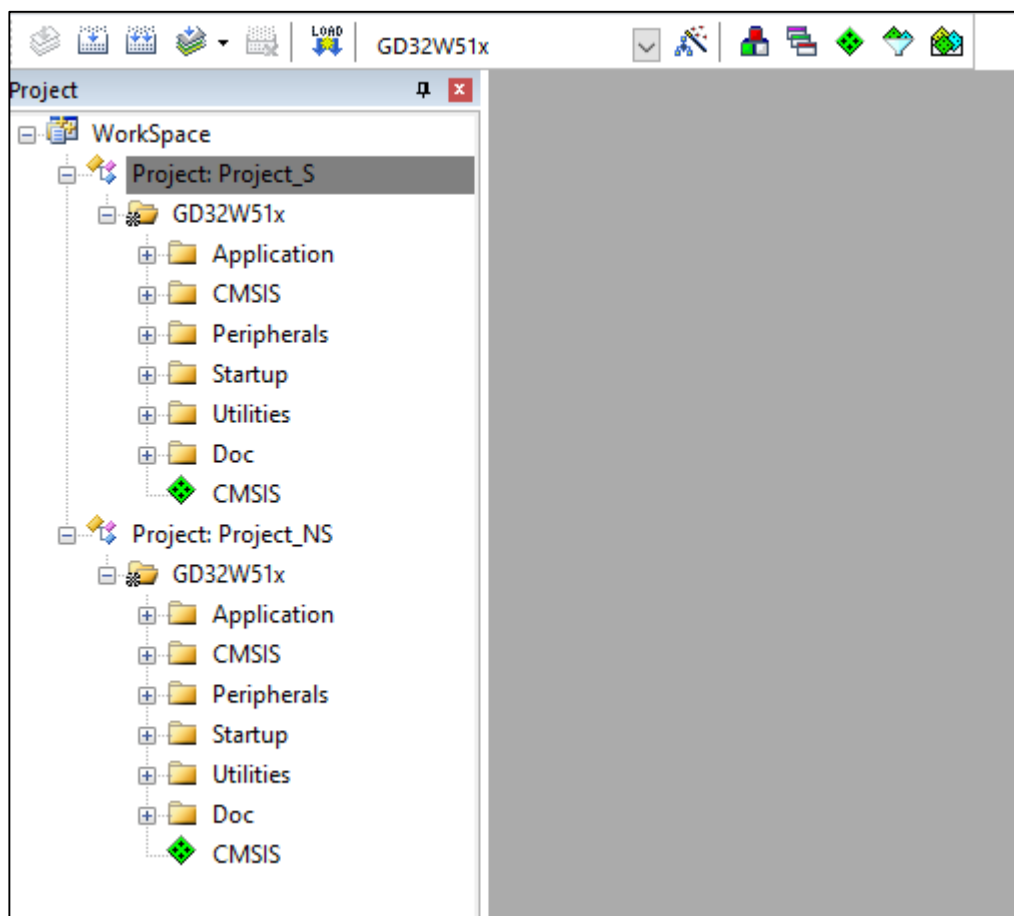
Figure 2-8. Copy the peripheral example files



Open a workspace

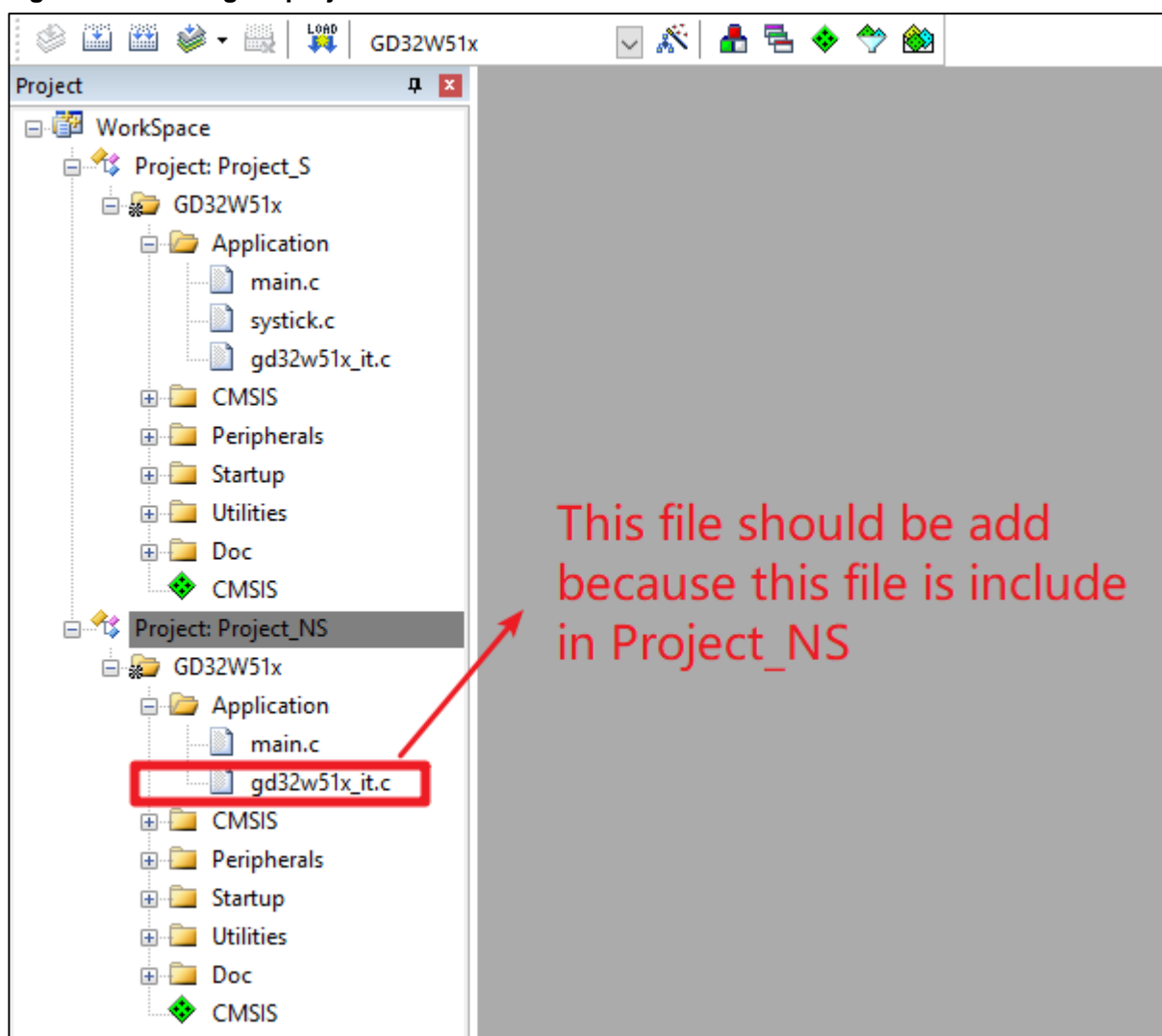
GD provides workspace in Keil and IAR, users can open workspace in different IDEs according to their need, such as "Keil_project" workspace, open \Template_TrustZone\Project.uvmpw, shown as below:

Figure 2-9. Open the workspace



The workspace include two projects, secure project and non-secure project. Because different module and different functions adopt different files, users should add or delete the files in project according to the copied files, shown as below:

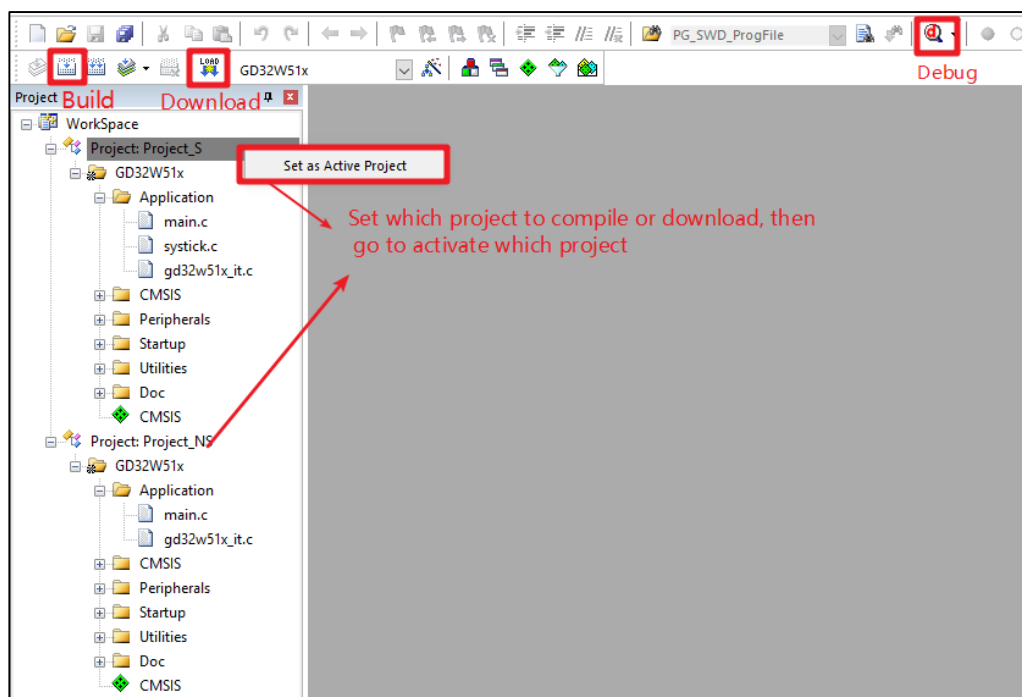
Figure 2-10. Configure project files



Compile-Debug-Download

First compile two projects separately, if there is no error, then select the right jumper cap according to the description of readme, download the two projects to the target board separately, 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 2-11. Compile-debug-download



2.1.5. Utilities Folder

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

- gd32w515p_eval.h are related header files of the evaluation board about running the firmware examples;
- gd32w515p_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
gd32w51x_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.
gd32w51x_it.h	Header file, including all the prototypes of interrupt service routines.
gd32w51x_it.c	Source files about interrupt 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.
gd32w51x_xxx.h	The header file of peripheral PPP, including functions about peripheral PPP, and the variables used for functions.
gd32w51x_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.
partition_gd32w51x.h	The header file configures SAU/IDAU and secure/non-secure interrupts attributes;

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 function
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)
ADC_WDHT	Watchdog high threshold register
ADC_WDLT	Watchdog low threshold register

Registers	Descriptions
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 (x=0..3)	Inserted data register x(x=0..3)
ADC_RDATA	Regular data register
ADC_OVSAMPCTL	Oversample control register
ADC_CCTL	Common 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 ADC peripheral
adc_clock_config	configure the ADC clock
adc_enable	enable ADC interface
adc_disable	disable ADC interface
adc_dma_mode_enable	enable DMA request
adc_dma_mode_disable	disable DMA request
adc_dma_request_after_last_enable	when DMA=1, the DMA engine issues a request at end of each regular conversion
adc_dma_request_after_last_disable	the DMA engine is disabled after the end of transfer signal from DMA controller is detected
adc_discontinuous_mode_config	configure ADC discontinuous mode
adc_special_function_config	enable or disable ADC special function
adc_channel_9_to_11	configure temperature sensor and internal reference voltage channel or VBAT channel function
adc_data_alignment_config	configure ADC data alignment
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_config	enable ADC external trigger
adc_external_trigger_source_config	configure ADC external trigger source
adc_software_trigger_enable	enable ADC software trigger
adc_end_of_conversion_config	configure end of conversion mode
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_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_oversample_mode_config	configure ADC oversample mode
adc_oversample_mode_enable	enable ADC oversample mode
adc_oversample_mode_disable	disable ADC oversample mode
adc_flag_get	get the ADC flag bits
adc_flag_clear	clear the ADC flag bits
adc_interrupt_enable	enable ADC interrupt
adc_interrupt_disable	disable ADC interrupt
adc_interrupt_flag_get	get the ADC interrupt bits
adc_interrupt_flag_clear	clear the ADC flag
adc_regular_software_startconv_flag_get	get the bit state of ADC software start conversion
adc_inserted_software_startconv_flag_get	get the bit state of ADC software inserted channel start conversion

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(void);
Function descriptions	reset ADC peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* reset ADC */

adc_deinit();

```

adc_clock_config

The description of adc_clock_config is shown as below:

Table 3-5. Function adc_clock_config

Function name	adc_clock_config
Function prototype	void adc_clock_config(uint32_t prescaler);
Function descriptions	configure the ADC clock
Precondition	-
The called functions	-
Input parameter{in}	
prescaler	configure ADC prescaler ratio
ADC_ADCCK_PCLK2_DIV2	PCLK2 div2
ADC_ADCCK_PCLK2_DIV4	PCLK2 div4
ADC_ADCCK_PCLK2_DIV6	PCLK2 div6
ADC_ADCCK_PCLK2_DIV8	PCLK2 div8
ADC_ADCCK_HCLK_DIV5	HCLK div5
ADC_ADCCK_HCLK_DIV6	HCLK div6
ADC_ADCCK_HCLK_DIV10	HCLK div10
ADC_ADCCK_HCLK_DIV20	HCLK div20
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the ADC clock */
adc_clock_config(ADC_ADCCK_PCLK2_DIV8);
```

adc_enable

The description of adc_enable is shown as below:

Table 3-6. Function adc_enable

Function name	adc_enable
Function prototype	void adc_enable(void);

Function descriptions	enable ADC interface
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

adc_disable

The description of adc_disable is shown as below:

Table 3-7. Function adc_disable

Function name	adc_disable
Function prototype	void adc_disable(void);
Function descriptions	disable ADC interface
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC */
adc_disable();
```

adc_dma_mode_enable

The description of adc_dma_mode_enable is shown as below:

Table 3-8. Function adc_dma_mode_enable

Function name	adc_dma_mode_enable
Function prototype	void adc_dma_mode_enable(void);
Function descriptions	enable ADC DMA request
Precondition	-

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

Example:

```
/* enable ADC DMA request */
```

```
adc_dma_mode_enable();
```

adc_dma_mode_disable

The description of adc_dma_mode_disable is shown as below:

Table 3-9. Function adc_dma_mode_disable

Function name	adc_dma_mode_disable
Function prototype	void adc_dma_mode_disable(void);
Function descriptions	disable ADC DMA request
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC DMA request */
```

```
adc_dma_mode_disable();
```

adc_dma_request_after_last_enable

The description of adc_dma_request_after_last_enable is shown as below:

Table 3-10. Function adc_dma_request_after_last_enable

Function name	adc_dma_request_after_last_enable
Function prototype	void adc_dma_request_after_last_enable(void);
Function descriptions	when DMA=1, the DMA engine issues a request at end of each regular conversion
Precondition	-
The called functions	-

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

Example:

/* when DMA=1, the DMA engine issues a request at end of each regular conversion */

```
adc_dma_request_after_last_enable();
```

adc_dma_request_after_last_disable

The description of adc_dma_request_after_last_disable is shown as below:

Table 3-11. Function adc_dma_request_after_last_disable

Function name	adc_dma_request_after_last_disable
Function prototype	void adc_dma_request_after_last_disable(void);
Function descriptions	the DMA engine is disabled after the end of transfer signal from DMA controller is detected
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* the DMA engine is disabled after the end of transfer signal from DMA is detected */

```
adc_dma_request_after_last_enable();
```

adc_discontinuous_mode_config

The description of adc_discontinuous_mode_config is shown as below:

Table 3-12. Function adc_discontinuous_mode_config

Function name	adc_discontinuous_mode_config
Function prototype	void adc_discontinuous_mode_config(uint8_t adc_channel_group, uint8_t length);
Function descriptions	configure ADC discontinuous mode
Precondition	-
The called functions	-

Input parameter{in}	
adc_channel_group	select the channel group
<i>ADC_REGULAR_CHANNEL</i>	regular channel group
<i>ADC_INSERTED_CHANNEL</i>	inserted channel group
<i>ADC_CHANNEL_DISCON_DISABLE</i>	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 ADC regular channel group discontinuous mode */
```

```
adc_discontinuous_mode_config(ADC_REGULAR_CHANNEL, 6);
```

adc_special_function_config

The description of adc_special_function_config is shown as below:

Table 3-13. Function adc_special_function_config

Function name	adc_special_function_config
Function prototype	void adc_special_function_config(uint32_t function, ControlStatus newvalue);
Function descriptions	enable or disable ADC special function
Precondition	-
The called functions	-
Input parameter{in}	
function	the function to config
<i>ADC_SCAN_MODE</i>	scan mode select
<i>ADC_INSERTED_CHANNEL_AUTO</i>	inserted channel group convert automatically
<i>ADC_CONTINUOUS_MODE</i>	continuous mode select
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* enable ADC scan mode */
```

```
adc_special_function_config(ADC_SCAN_MODE, ENABLE);
```

adc_channel_9_to_11

The description of adc_channel_9_to_11 is shown as below:

Table 3-14. Function adc_channel_9_to_11

Function name	adc_channel_9_to_11
Function prototype	void adc_channel_9_to_11(uint32_t function, ControlStatus newvalue);
Function descriptions	configure temperature sensor and internal reference voltage channel or VBAT channel function
Precondition	-
The called functions	-
Input parameter{in}	
function	temperature sensor and internal reference voltage channel or VBAT channel
ADC_VBAT_CHANNEL_SWITCH	channel 11 (1/4 voltate of external battery) switch of ADC
ADC_TEMP_VREF_CHANNEL_SWITCH	channel 9 (temperature sensor) and 10 (internal reference voltage) switch of ADC
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure VBAT channel function */
```

```
adc_channel_9_to_11 (ADC_VBAT_CHANNEL_SWITCH, ENABLE);
```

adc_data_alignment_config

The description of adc_data_alignment_config is shown as below:

Table 3-15. Function `adc_data_alignment_config`

Function name	<code>adc_data_alignment_config</code>
Function prototype	<code>void adc_data_alignment_config(uint32_t data_alignment);</code>
Function descriptions	configure ADC data alignment
Precondition	-
The called functions	-
Input parameter{in}	
data_alignment	data alignment select
<code>ADC_DATAALIGN_RIGHT</code>	right alignment
<code>ADC_DATAALIGN_LEFT</code>	left alignment
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC data alignment */
```

```
adc_data_alignment_config(ADC_DATAALIGN_RIGHT);
```

adc_channel_length_config

The description of `adc_channel_length_config` is shown as below:

Table 3-16. Function `adc_channel_length_config`

Function name	<code>adc_channel_length_config</code>
Function prototype	<code>void adc_channel_length_config(uint8_t adc_channel_group, uint32_t length);</code>
Function descriptions	configure the length of regular channel group or inserted channel group
Precondition	-
The called functions	-
Input parameter{in}	
adc_channel_group	select the channel group
<code>ADC_REGULAR_CHANNEL</code>	regular channel group
<code>ADC_INSERTED_CHANNEL</code>	inserted channel group
Input parameter{in}	
length	the length of the channel, regular channel 1-9, inserted channel 1-4
Output parameter{out}	
-	-
Return value	

Example:

```
/* configure the length of ADC regular channel */
```

```
adc_channel_length_config(ADC_REGULAR_CHANNEL, 4);
```

adc_regular_channel_config

The description of adc_regular_channel_config is shown as below:

Table 3-17. Function adc_regular_channel_config

Function name	adc_regular_channel_config
Function prototype	void adc_regular_channel_config(uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	configure ADC regular channel
Precondition	-
The called functions	-
Input parameter{in}	
rank	the regular group sequence rank, this parameter must be between 0 to 8
Input parameter{in}	
adc_channel	the selected ADC channel
ADC_CHANNEL_x (x=0..11)	ADC Channelx (x=0..11)
Input parameter{in}	
sample_time	the sample time value
ADC_SAMPLETIME_2	2 cycles
ADC_SAMPLETIME_15	15 cycles
ADC_SAMPLETIME_28	28 cycles
ADC_SAMPLETIME_56	56 cycles
ADC_SAMPLETIME_84	84 cycles
ADC_SAMPLETIME_112	112 cycles
ADC_SAMPLETIME_144	144 cycles
ADC_SAMPLETIME_480	480 cycles
Output parameter{out}	
-	-
Return value	

Example:

```
/* configure ADC regular channel */
```

```
adc_regular_channel_config(1, ADC_CHANNEL_0, ADC_SAMPLETIME_15);
```

adc_inserted_channel_config

The description of adc_inserted_channel_config is shown as below:

Table 3-18. Function adc_inserted_channel_config

Function name	adc_inserted_channel_config
Function prototype	void adc_inserted_channel_config(uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	configure ADC inserted channel
Precondition	-
The called functions	-
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..11)	ADC Channelx (x=0..11)
Input parameter{in}	
sample_time	the sample time value
ADC_SAMPLETIME_2	2 cycles
ADC_SAMPLETIME_15	15 cycles
ADC_SAMPLETIME_28	28 cycles
ADC_SAMPLETIME_56	56 cycles
ADC_SAMPLETIME_84	84 cycles
ADC_SAMPLETIME_112	112 cycles
ADC_SAMPLETIME_144	144 cycles
ADC_SAMPLETIME_480	480 cycles
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* configure ADC inserted channel */
```

```
adc_inserted_channel_config(1, ADC_CHANNEL_0, ADC_SAMPLETIME_15);
```

adc_inserted_channel_offset_config

The description of adc_inserted_channel_offset_config is shown as below:

Table 3-19. Function adc_inserted_channel_offset_config

Function name	adc_inserted_channel_offset_config
Function prototype	void adc_inserted_channel_offset_config(uint8_t inserted_channel, uint16_t offset);
Function descriptions	configure ADC inserted channel offset
Precondition	-
The called functions	-
Input parameter{in}	
inserted_channel	insert channel select
<i>ADC_INSERTED_CHANNEL_x(x=0..3)</i>	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 ADC inserted channel offset */
```

```
adc_inserted_channel_offset_config(ADC_INSERTED_CHANNEL_0, 100);
```

adc_external_trigger_config

The description of adc_external_trigger_config is shown as below:

Table 3-20. Function adc_external_trigger_config

Function name	adc_external_trigger_config
Function prototype	void adc_external_trigger_config(uint8_t channel_group, uint32_t trigger_mode);
Function descriptions	configure ADC external trigger
Precondition	-
The called functions	-
Input parameter{in}	

adc_channel_group	select the channel group
<i>ADC_REGULAR_CHANNEL</i>	regular channel group
<i>ADC_INSERTED_CHANNEL</i>	inserted channel group
Input parameter{in}	
trigger_mode	external trigger mode
<i>EXTERNAL_TRIGGER_DISABLE</i>	external trigger disable
<i>EXTERNAL_TRIGGER_RISING</i>	rising edge of external trigger
<i>EXTERNAL_TRIGGER_FALLING</i>	falling edge of external trigger
<i>EXTERNAL_TRIGGER_RISING_FALLING</i>	rising and falling edge of external trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC inserted channel group external trigger */
```

```
adc_external_trigger_config(ADC_INSERTED_CHANNEL,  
EXTERNAL_TRIGGER_RISING);
```

adc_external_trigger_source_config

The description of adc_external_trigger_source_config is shown as below:

Table 3-21. Function adc_external_trigger_source_config

Function name	adc_external_trigger_source_config
Function prototype	void adc_external_trigger_source_config(uint8_t channel_group, uint32_t external_trigger_source);
Function descriptions	configure ADC external trigger source
Precondition	-
The called functions	-
Input parameter{in}	
adc_channel_group	select the channel group
<i>ADC_REGULAR_CHANNEL</i>	regular channel group
<i>ADC_INSERTED_CHANNEL</i>	inserted channel group
Input parameter{in}	

external_trigger_ source	regular or inserted group trigger source
<i>ADC_EXTTRIG_ REGULAR_T0_CH0</i>	TIMER0 CH0 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T0_CH1</i>	TIMER0 CH1 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T0_CH2</i>	TIMER0 CH2 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T1_CH1</i>	TIMER1 CH1 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T1_CH2</i>	TIMER1 CH2 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T1_CH3</i>	TIMER1 CH3 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T1_TRGO</i>	TIMER1 TRGO event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T2_CH0</i>	TIMER2 CH0 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T2_TRGO</i>	TIMER2 TRGO event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T3_CH3</i>	TIMER3 CH3 event select for regular channel
<i>ADC_EXTTRIG_REGU LAR_T4_CH0</i>	TIMER4 CH0 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T4_CH1</i>	TIMER4 CH1 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_T4_CH2</i>	TIMER4 CH2 event select for regular channel
<i>ADC_EXTTRIG_ REGULAR_EXTI_11</i>	external interrupt line 11 for regular channel
<i>ADC_EXTTRIG_ INSERTED_T0_CH3</i>	TIMER0 CH3 event select for inserted channel
<i>ADC_EXTTRIG_ INSERTED_T0_TRGO</i>	TIMER0 TRGO event select for inserted channel
<i>ADC_EXTTRIG_ INSERTED_T1_CH0</i>	TIMER1 CH0 event select for inserted channel
<i>ADC_EXTTRIG_ INSERTED_T1_TRGO</i>	TIMER1 TRGO event select for inserted channel
<i>ADC_EXTTRIG_ INSERTED_T2_CH1</i>	TIMER2 CH1 event select for inserted channel
<i>ADC_EXTTRIG_ INSERTED_T2_CH3</i>	TIMER3 CH3 event select for inserted channel
<i>ADC_EXTTRIG_ INSERTED_T3_CH0</i>	TIMER3 CH0 event select for inserted channel

<i>INSERTED_T3_CH0</i>	
<i>ADC_EXTTRIG_INSERTED_T3_CH1</i>	TIMER3 CH1 event select for inserted channel
<i>ADC_EXTTRIG_INSERTED_T3_CH2</i>	TIMER3 CH2 event select for inserted channel
<i>ADC_EXTTRIG_INSERTED_T3_TRGO</i>	TIMER3 TRGO event select for inserted channel
<i>ADC_EXTTRIG_INSERTED_T4_CH3</i>	TIMER4 CH3 event select for inserted channel
<i>ADC_EXTTRIG_INSERTED_T4_TRGO</i>	TIMER4 TRGO event select for inserted channel
<i>ADC_EXTTRIG_INSERTED_EXTI_15</i>	external interrupt line 15 for inserted channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC regular channel external trigger source */
```

```
adc_external_trigger_source_config (ADC_REGULAR_CHANNEL,  
ADC_EXTTRIG_REGULAR_T0_CH0);
```

adc_software_trigger_enable

The description of `adc_software_trigger_enable` is shown as below:

Table 3-22. Function `adc_software_trigger_enable`

Function name	<code>adc_software_trigger_enable</code>
Function prototype	<code>void adc_software_trigger_enable(uint8_t adc_channel_group);</code>
Function descriptions	enable ADC software trigger
Precondition	-
The called functions	-
Input parameter{in}	
adc_channel_group	select the channel group
<i>ADC_REGULAR_CHANNEL</i>	regular channel group
<i>ADC_INSERTED_CHANNEL</i>	inserted channel group
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC regular channel group software trigger */
```

```
adc_software_trigger_enable(ADC_REGULAR_CHANNEL);
```

adc_end_of_conversion_config

The description of adc_end_of_conversion_config is shown as below:

Table 3-23. Function adc_end_of_conversion_config

Function name	adc_end_of_conversion_config
Function prototype	void adc_end_of_conversion_config(uint8_t end_selection);
Function descriptions	configure end of conversion mode
Precondition	-
The called functions	-
Input parameter{in}	
end_selection	end of conversion mode
ADC_EOC_SET_SEQUENCE	only at the end of a sequence of regular conversions, the EOC bit is set. overflow detection is disabled unless DMA=1.
ADC_EOC_SET_CONVERSION	at the end of each regular conversion, the EOC bit is set. Overflow is detected automatically.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure end of conversion mode */
```

```
adc_end_of_conversion_config(ADC_EOC_SET_SEQUENCE);
```

adc_regular_data_read

The description of adc_regular_data_read is shown as below:

Table 3-24. Function adc_regular_data_read

Function name	adc_regular_data_read
Function prototype	uint16_t adc_regular_data_read(void);
Function descriptions	read ADC regular group data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
uint16_t	ADC conversion value (0~0xFFFF)

Example:

```
/* read ADC regular group data register */
```

```
uint16_t adc_value = 0;
```

```
adc_value = adc_regular_data_read();
```

adc_inserted_data_read

The description of `adc_inserted_data_read` is shown as below:

Table 3-25. Function `adc_inserted_data_read`

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

Example:

```
/* read ADC inserted group data register */
```

```
uint16_t adc_value = 0;
```

```
adc_value = adc_inserted_data_read (ADC_INSERTED_CHANNEL_0);
```

adc_watchdog_single_channel_enable

The description of `adc_watchdog_single_channel_enable` is shown as below:

Table 3-26. Function `adc_watchdog_single_channel_enable`

Function name	<code>adc_watchdog_single_channel_enable</code>
Function prototype	<code>void adc_watchdog_single_channel_enable(uint8_t adc_channel);</code>
Function descriptions	configure ADC analog watchdog single channel
Precondition	-
The called functions	-
Input parameter{in}	

adc_channel	the selected ADC channel
<i>ADC_CHANNEL_x</i> (x=0..11)	ADC channelx(x=0..11)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC analog watchdog single channel */
```

```
adc_watchdog_single_channel_enable(ADC_CHANNEL_1);
```

adc_watchdog_group_channel_enable

The description of adc_watchdog_group_channel_enable is shown as below:

Table 3-27. Function adc_watchdog_group_channel_enable

Function name	adc_watchdog_group_channel_enable
Function prototype	void adc_watchdog_group_channel_enable(uint8_t adc_channel_group);
Function descriptions	configure ADC analog watchdog group channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_channel_group	the channel group use analog watchdog
<i>ADC_REGULAR_CHANNEL</i>	regular channel group
<i>ADC_INSERTED_CHANNEL</i>	inserted channel group
<i>ADC_REGULAR_INSERTED_CHANNEL</i>	both regular and inserted group
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC analog watchdog group channel */
```

```
adc_watchdog_group_channel_enable(ADC_REGULAR_CHANNEL);
```

adc_watchdog_disable

The description of adc_watchdog_disable is shown as below:

Table 3-28. Function `adc_watchdog_disable`

Function name	<code>adc_watchdog_disable</code>
Function prototype	<code>void adc_watchdog_disable(void);</code>
Function descriptions	disable ADC analog watchdog
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

`adc_watchdog_threshold_config`

The description of `adc_watchdog_threshold_config` is shown as below:

Table 3-29. Function `adc_watchdog_threshold_config`

Function name	<code>adc_watchdog_threshold_config</code>
Function prototype	<code>void adc_watchdog_threshold_config(uint16_t low_threshold , uint16_t high_threshold);</code>
Function descriptions	configure ADC analog watchdog threshold
Precondition	-
The called functions	-
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 ADC analog watchdog threshold */
adc_watchdog_threshold_config( 0x0400, 0x0A00);
```

adc_oversample_mode_config

The description of adc_oversample_mode_config is shown as below:

Table 3-30. Function adc_oversample_mode_config

Function name	adc_oversample_mode_config
Function prototype	void adc_oversample_mode_config(uint32_t mode, uint16_t shift, uint8_t ratio);
Function descriptions	configure ADC oversample mode
Precondition	-
The called functions	-
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 ADC oversample mode: 16 times sample, 4 bits shift */
```

```
adc_oversample_mode_config(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 3-31. Function adc_oversample_mode_enable

Function name	adc_oversample_mode_enable
Function prototype	void adc_oversample_mode_enable(void);
Function descriptions	enable ADC oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC oversample mode */
```

```
adc_oversample_mode_enable();
```

adc_oversample_mode_disable

The description of adc_oversample_mode_disable is shown as below:

Table 3-32. Function adc_oversample_mode_disable

Function name	adc_oversample_mode_disable
Function prototype	void adc_oversample_mode_disable(void);
Function descriptions	disable ADC oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

adc_flag_get

The description of adc_flag_get is shown as below:

Table 3-33. Function adc_flag_get

Function name	adc_flag_get
Function prototype	FlagStatus adc_flag_get(uint32_t flag);
Function descriptions	get the ADC flag
Precondition	-
The called functions	-
Input parameter{in}	
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
ADC_FLAG_ROVF	regular data register overflow flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ADC analog watchdog flag bits */

FlagStatus flag_value;

flag_value = adc_flag_get(ADC_FLAG_WDE);
```

adc_flag_clear

The description of adc_flag_clear is shown as below:

Table 3-34. Function adc_flag_clear

Function name	adc_flag_clear
Function prototype	void adc_flag_clear(uint32_t flag);
Function descriptions	clear the ADC flag bits
Precondition	-
The called functions	-
Input parameter{in}	
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
ADC_FLAG_ROVF	regular data register overflow flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the ADC analog watchdog flag bits */

adc_flag_clear(ADC_FLAG_WDE);
```

adc_interrupt_enable

The description of adc_interrupt_enable is shown as below:

Table 3-35. Function adc_interrupt_enable

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

interrupt	the adc interrupt
<i>ADC_INT_WDE</i>	analog watchdog interrupt
<i>ADC_INT_EOC</i>	end of group conversion interrupt
<i>ADC_INT_EOIC</i>	end of inserted group conversion interrupt
<i>ADC_INT_ROVF</i>	regular data register overflow interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC analog watchdog interrupt */
```

```
adc_interrupt_enable(ADC_INT_WDE);
```

adc_interrupt_disable

The description of adc_interrupt_disable is shown as below:

Table 3-36. Function adc_interrupt_disable

Function name	adc_interrupt_disable
Function prototype	void adc_interrupt_disable(uint32_t interrupt);
Function descriptions	disable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the adc interrupt
<i>ADC_INT_WDE</i>	analog watchdog interrupt
<i>ADC_INT_EOC</i>	end of group conversion interrupt
<i>ADC_INT_EOIC</i>	end of inserted group conversion interrupt
<i>ADC_INT_ROVF</i>	regular data register overflow interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC analog watchdog interrupt */
```

```
adc_interrupt_disable(ADC_INT_WDE);
```

adc_interrupt_flag_get

The description of adc_interrupt_flag_get is shown as below:

Table 3-37. Function `adc_interrupt_flag_get`

Function name	<code>adc_interrupt_flag_get</code>
Function prototype	<code>FlagStatus adc_interrupt_flag_get(uint32_t int_flag);</code>
Function descriptions	get the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	the adc interrupt bits
<code>ADC_INT_FLAG_WDE</code>	analog watchdog interrupt
<code>ADC_INT_FLAG_EOC</code>	end of group conversion interrupt
<code>ADC_INT_FLAG_EOIC</code>	end of inserted group conversion interrupt
<code>ADC_INT_FLAG_ROVF</code>	regular data register overflow interrupt
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ADC analog watchdog interrupt bits*/
```

```
FlagStatus flag_value;
```

```
flag_value = adc_interrupt_flag_get(ADC_INT_FLAG_WDE);
```

adc_interrupt_flag_clear

The description of `adc_interrupt_flag_clear` is shown as below:

Table 3-38. Function `adc_interrupt_flag_clear`

Function name	<code>adc_interrupt_flag_clear</code>
Function prototype	<code>void adc_interrupt_flag_clear(uint32_t int_flag);</code>
Function descriptions	clear the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	the adc interrupt bits
<code>ADC_INT_FLAG_WDE</code>	analog watchdog interrupt
<code>ADC_INT_FLAG_EOC</code>	end of group conversion interrupt
<code>ADC_INT_FLAG_EOIC</code>	end of inserted group conversion interrupt
<code>ADC_INT_FLAG_ROVF</code>	regular data register overflow interrupt
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* clear the ADC analog watchdog interrupt bits*/
adc_interrupt_flag_clear(ADC_INT_FLAG_WDE);
```

adc_regular_software_startconv_flag_get

The description of adc_regular_software_startconv_flag_get is shown as below:

Table 3-39. Function adc_interrupt_flag_get

Function name	adc_regular_software_startconv_flag_get
Function prototype	FlagStatus adc_regular_software_startconv_flag_get(void);
Function descriptions	get the bit state of ADC software start conversion
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the bit state of ADC software start conversion */
FlagStatus flag_value;
flag_value = adc_regular_software_startconv_flag_get();
```

adc_inserted_software_startconv_flag_get

The description of adc_inserted_software_startconv_flag_get is shown as below:

Table 3-40. Function adc_inserted_software_startconv_flag_get

Function name	adc_inserted_software_startconv_flag_get
Function prototype	FlagStatus adc_inserted_software_startconv_flag_get(void);
Function descriptions	get the bit state of ADC software inserted channel start conversion
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
FlagStatus	SET or RESET

Example:

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

```
FlagStatus flag_value;
```

```
flag_value = adc_inserted_software_startconv_flag_get ();
```

3.3. CAU

The Cryptographic Acceleration Unit supports acceleration of DES, Triple-DES or AES (128,192, or 256) algorithms. The CAU registers are listed in chapter [3.3.1](#) the CAU firmware functions are introduced in chapter [3.3.2](#)

3.3.1. Descriptions of Peripheral registers

CAU registers are listed in the table shown as below:

Table 3-41. CAU Registers

Registers	Descriptions
CAU_CTL	control register
CAU_STAT0	status register 0
CAU_DI	data input register
CAU_DO	data output register
CAU_DMAEN	DMA enable register
CAU_INTEN	interrupt enable register
CAU_STAT1	status register 1
CAU_INTF	interrupt flag register
CAU_KEY0H	key 0 high register
CAU_KEY0L	key 0 low register
CAU_KEY1H	key 1 high register
CAU_KEY1L	key 1 low register
CAU_KEY2H	key 2 high register
CAU_KEY2L	key 2 low register
CAU_KEY3H	key 3 high register
CAU_KEY3L	key 3 low register
CAU_IV0H	initial vector 0 high register
CAU_IV0L	initial vector 0 low register
CAU_IV1H	initial vector 1 high register
CAU_IV1L	initial vector 1 low register
CAU_GCMCCMCT	GCM or CCM mode context switch register x

Registers	Descriptions
XSx (x = 0..7)	
CAU_GCMCTXSx (x = 0..7)	GCM mode context switch register x

3.3.2. Descriptions of Peripheral functions

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

Table 3-42. CAU firmware function

Function name	Function description
cau_deinit	reset the CAU peripheral
cau_struct_para_init	initialize the CAU encrypt and decrypt parameter struct with the default values
cau_key_struct_para_init	initialize the key parameter struct with the default values
cau_iv_struct_para_init	initialize the vectors parameter struct with the default values
cau_context_struct_para_init	initialize the context parameter struct with the default values
cau_enable	enable the CAU peripheral
cau_disable	disable the CAU peripheral
cau_dma_enable	enable the CAU DMA interface
cau_dma_disable	disable the CAU DMA interface
cau_init	initialize the CAU peripheral
cau_aes_keysize_config	configure key size if use AES algorithm
cau_key_init	initialize the key parameters
cau_iv_init	initialize the vectors parameters
cau_phase_config	configure phase
cau_fifo_flush	flush the IN and OUT FIFOs
cau_enable_state_get	return whether CAU peripheral is enabled or disabled
cau_data_write	write data to the IN FIFO
cau_data_read	return the last data entered into the output FIFO
cau_context_save	save context before context switching
cau_context_restore	restore context after context switching
cau_aes_ecb	encrypt and decrypt using AES in ECB mode
cau_aes_cbc	encrypt and decrypt using AES in CBC mode
cau_aes_ctr	encrypt and decrypt using AES in CTR mode
cau_aes_cfb	encrypt and decrypt using AES in CFB mode
cau_aes_ofb	encrypt and decrypt using AES in OFB mode
cau_aes_gcm	encrypt and decrypt using AES in GCM mode
cau_aes_ccm	encrypt and decrypt using AES in CCM mode
cau_tdes_ecb	encrypt and decrypt using TDES in ECB mode
cau_tdes_cbc	encrypt and decrypt using TDES in CBC mode
cau_des_ecb	encrypt and decrypt using DES in ECB mode
cau_des_cbc	encrypt and decrypt using DES in CBC mode

Function name	Function description
cau_interrupt_enable	enable the CAU interrupts
cau_interrupt_disable	disable the CAU interrupts
cau_interrupt_flag_get	get the interrupt flag
cau_flag_get	get the CAU flag status

Structure cau_key_parameter_struct

Table 3-43. Structure cau_key_parameter_struct

Member name	Function description
key_0_high	key 0 high
key_0_low	key 0 low
key_1_high	key 1 high
key_1_low	key 1 low
key_2_high	key 2 high
key_2_low	key 2 low
key_3_high	key 3 high
key_3_low	key 3 low

Structure cau_iv_parameter_struct

Table 3-44. Structure cau_iv_parameter_struct

Member name	Function description
iv_0_high	init vector 0 high
iv_0_low	init vector 0 low
iv_1_high	init vector 1 high
iv_1_low	init vector 1 low

Structure cau_context_parameter_struct

Table 3-45. Structure cau_context_parameter_struct

Member name	Function description
ctl_config	current configuration
iv_0_high	init vector 0 high
iv_0_low	init vector 0 low
iv_1_high	init vector 1 high
iv_1_low	init vector 1 low
key_0_high	key 0 high
key_0_low	key 0 low
key_1_high	key 1 high
key_1_low	key 1 low
key_2_high	key 2 high
key_2_low	key 2 low
key_3_high	key 3 high

key_3_low	key 3 low
gcmccmctxs[8]	GCM or CCM mode context switch
gcmctxs[8]	GCM mode context switch

Structure cau_parameter_struct

Table 3-46. Structure cau_parameter_struct

Member name	Function description
alg_dir	algorithm directory
*key	key
key_size	key size in bytes
*iv	initialization vector
iv_size	iv size in bytes
*input	input data
in_length	input data length in bytes
*aad	additional authentication data
aad_size	aad size

cau_deinit

The description of cau_deinit is shown as below:

Table 3-47. Function cau_deinit

Function name	cau_deinit
Function prototype	void cau_deinit(void)
Function descriptions	reset the CAU peripheral
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the CAU peripheral */
```

```
cau_deinit( );
```

cau_struct_para_init

The description of cau_struct_para_init is shown as below:

Table 3-48. Function cau_struct_para_init

Function name	cau_struct_para_init
---------------	----------------------

Function prototype	void cau_struct_para_init(cau_parameter_struct *cau_parameter)
Function descriptions	initialize the CAU encrypt and decrypt parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Return value	
-	-

Example:

```
cau_parameter_struct text;
```

```
/* initialize CAU encrypt and decrypt parameter struct */
```

```
cau_struct_para_init(&text);
```

cau_key_struct_para_init

The description of cau_key_struct_para_init is shown as below:

Table 3-49. Function cau_key_struct_para_init

Function name	cau_key_struct_para_init
Function prototype	void cau_key_struct_para_init(cau_key_parameter_struct *key_initpara)
Function descriptions	initialize the key parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
key_initpara	the key parameters, refer to structure Table 3-43. Structure cau_key_parameter_struct
Return value	
-	-

Example:

```
/* initialize the key parameter struct */
```

```
cau_key_parameter_struct key_initpara;
```

```
cau_key_struct_para_init(&key_initpara);
```

cau_iv_struct_para_init

The description of cau_iv_struct_para_init is shown as below:

Table 3-50. Function cau_iv_struct_para_init

Function name	cau_iv_struct_para_init
Function prototype	void cau_iv_struct_para_init(cau_iv_parameter_struct *iv_initpara)
Function descriptions	initialize the vectors parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
iv_initpara	the vectors parameter, refer to structure Table 3-44. Structure cau_iv_parameter_struct
Return value	
-	-

Example:

```
/* initialize the vectors parameter struct */
cau_iv_parameter_struct iv_initpara;
cau_iv_struct_para_init(&iv_initpara);
```

cau_context_struct_para_init

The description of cau_context_struct_para_init is shown as below:

Table 3-51. Function cau_context_struct_para_init

Function name	cau_context_struct_para_init
Function prototype	void cau_context_struct_para_init(cau_context_parameter_struct *cau_context)
Function descriptions	initialize the context parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
cau_context	the context parameter, refer to structure Table 3-45. Structure cau_context_parameter_struct
Return value	
-	-

Example:

```
/* initialize the context parameter struct */
```

```
cau_context_parameter_struct context_initpara;
```

```
cau_context_struct_para_init (&context_initpara);
```

cau_enable

The description of cau_enable is shown as below:

Table 3-52. Function cau_enable

Function name	cau_enable
Function prototype	void cau_enable(void);
Function descriptions	enable the CAU peripheral
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the CAU peripheral */
cau_enable();
```

cau_disable

The description of cau_disable is shown as below:

Table 3-53. Function cau_disable

Function name	cau_disable
Function prototype	void cau_disable(void);
Function descriptions	disable the CAU peripheral
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CAU peripheral */
```

```
cau_disable();
```

cau_dma_enable

The description of cau_dma_enable is shown as below:

Table 3-54. Function cau_dma_enable

Function name	cau_dma_enable
Function prototype	void cau_dma_enable(uint32_t dma_req);
Function descriptions	enable the CAU DMA interface
Precondition	-
The called functions	-
Input parameter{in}	
dma_req	specify the CAU DMA transfer request to be enabled
CAU_DMA_INFIFO	DMA for incoming(Rx) data transfer
CAU_DMA_OUTFIFO	DMA for outgoing(Tx) data transfer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the CAU DMA interface */
cau_dma_enable(CAU_DMA_INFIFO);
```

cau_dma_disable

The description of cau_dma_disable is shown as below:

Table 3-55. Function cau_dma_disable

Function name	cau_dma_disable
Function prototype	void cau_dma_disable(uint32_t dma_req);
Function descriptions	disable the CAU DMA interface
Precondition	-
The called functions	-
Input parameter{in}	
dma_req	specify the CAU DMA transfer request to be disabled
CAU_DMA_INFIFO	DMA for incoming(Rx) data transfer
CAU_DMA_OUTFIFO	DMA for outgoing(Tx) data transfer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CAU DMA interface */
```

```
cau_dma_disable(CAU_DMA_INFIFO);
```

cau_init

The description of cau_init is shown as below:

Table 3-56. Function cau_init

Function name	cau_init
Function prototype	void cau_init(uint32_t alg_dir, uint32_t algo_mode, uint32_t swapping)
Function descriptions	initialize the CAU peripheral
Precondition	-
The called functions	-
Input parameter{in}	
alg_dir	algorithm direction
CAU_ENCRYPT	encrypt
CAU_DECRYPT	decrypt
Input parameter{in}	
algo_mode	algorithm mode selection
CAU_MODE_TDES_ECB	TDES-ECB (3DES Electronic codebook)
CAU_MODE_TDES_CBC	TDES-CBC (3DES Cipher block chaining)
CAU_MODE_DES_ECB	DES-ECB (simple DES Electronic codebook)
CAU_MODE_DES_CBC	DES-CBC (simple DES Cipher block chaining)
CAU_MODE_AES_ECB	AES-ECB (AES Electronic codebook)
CAU_MODE_AES_CBC	AES-CBC (AES Cipher block chaining)
CAU_MODE_AES_CTR	AES-CTR (AES counter mode)
CAU_MODE_AES_KEY	AES decryption key preparation mode
CAU_MODE_AES_GCM	AES-GCM (AES Galois/counter mode)
CAU_MODE_AES_CCM	AES-CCM (AES combined cipher machine mode)
CAU_MODE_AES_CFB	AES-CFB (cipher feedback mode)
CAU_MODE_AES_OFB	AES-OFB (output feedback mode)

Input parameter{in}	
swapping	data swapping selection
CAU_SWAPPING_32BIT	no swapping
CAU_SWAPPING_16BIT	half-word swapping
CAU_SWAPPING_8BIT	bytes swapping
CAU_SWAPPING_1BIT	bit swapping
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the CAU peripheral */
cau_init(CAU_ENCRYPT, CAU_MODE_TDES_ECB, CAU_SWAPPING_32BIT);
```

cau_aes_keysize_config

The description of cau_aes_keysize_config is shown as below:

Table 3-57. Function cau_aes_keysize_config

Function name	cau_aes_keysize_config
Function prototype	void cau_aes_keysize_config(uint32_t key_size);
Function descriptions	configure key size if used AES algorithm
Precondition	-
The called functions	-
Input parameter{in}	
key_size	key length selection when aes mode
CAU_KEYSIZE_128BIT	128 bit key length
CAU_KEYSIZE_192BIT	192 bit key length
CAU_KEYSIZE_256BIT	256 bit key length
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure key size if used AES algorithm */
cau_aes_keysize_config(CAU_KEYSIZE_128BIT);
```

cau_key_init

The description of cau_key_init is shown as below:

Table 3-58. Function cau_key_init

Function name	cau_key_init
Function prototype	void cau_key_init(cau_key_parameter_struct* key_initpara);
Function descriptions	initialize the key parameters
Precondition	-
The called functions	-
Input parameter{in}	
key_initpara	the key parameters, refer to structure Table 3-43. Structure cau_key_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the key parameters */
cau_key_parameter_struct key_initpara;
cau_key_init (&key_initpara);
```

cau_iv_init

The description of cau_iv_init is shown as below:

Table 3-59. Function cau_iv_init

Function name	cau_iv_init
Function prototype	void cau_iv_init(cau_iv_parameter_struct* iv_initpara);
Function descriptions	initialize the vectors parameters
Precondition	-
The called functions	-
Input parameter{in}	
iv_initpara	the vectors parameters, refer to structure Table 3-44. Structure cau_iv_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the vectors parameters */
```

```
cau_iv_parameter_struct iv_initpara;
```

```
cau_iv_init(&iv_initpara);
```

cau_phase_config

The description of cau_phase_config is shown as below:

Table 3-60. Function cau_phase_config

Function name	cau_phase_config
Function prototype	void cau_phase_config(uint32_t phase)
Function descriptions	configure phase
Precondition	-
The called functions	-
Input parameter{in}	
phase	gcm or ccm phase
CAU_PREPARE_PHASE	prepare phase
CAU_AAD_PHASE	AAD phase
CAU_ENCRYPT_DECRYPT_PHASE	encryption/decryption phase
CAU_TAG_PHASE	tag phase
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select prepare phase */
```

```
cau_phase_config(CAU_PREPARE_PHASE);
```

cau_fifo_flush

The description of cau_fifo_flush is shown as below:

Table 3-61. Function cau_fifo_flush

Function name	cau_fifo_flush
Function prototype	void cau_fifo_flush(void);
Function descriptions	flush the IN and OUT FIFOs
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* flush the IN and OUT FIFOs */
```

```
cau_fifo_flush( );
```

cau_enable_state_get

The description of cau_enable_state_get is shown as below:

Table 3-62. Function cau_enable_state_get

Function name	cau_enable_state_get
Function prototype	ControlStatus cau_enable_state_get(void);
Function descriptions	return whether CAU peripheral is enabled or disabled
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ControlStatus	ENABLE or DISABLE

Example:

```
/* return whether CAU peripheral is enabled or disabled */
```

```
ControlStatus state = cau_enable_state_get();
```

cau_data_write

The description of cau_data_write is shown as below:

Table 3-63. Function cau_data_write

Function name	cau_data_write
Function prototype	void cau_data_write(uint32_t data);
Function descriptions	write data to the IN FIFO
Precondition	-
The called functions	-
Input parameter{in}	
data	data to write: 0 - 0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write data to the IN FIFO */

cau_data_write(0x10);
```

cau_data_read

The description of cau_data_read is shown as below:

Table 3-64. Function cau_data_read

Function name	cau_data_read
Function prototype	uint32_t cau_data_read(void);
Function descriptions	return the last data entered into the output FIFO
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	0x0 – 0xFFFFFFFF

Example:

```
/* return the last data entered into the output FIFO */

uint32_t data;

data = cau_data_read();
```

cau_context_save

The description of cau_context_save is shown as below:

Table 3-65. Function cau_context_save

Function name	cau_context_save
Function prototype	void cau_context_save(cau_context_parameter_struct *cau_context, cau_key_parameter_struct* key_initpara)
Function descriptions	save context before context switching
Precondition	-
The called functions	-
Input parameter{in}	
key_initpara	the key parameters, refer to structure Table 3-43. Structure cau_key_parameter_struct
Output parameter{out}	
cau_context	the context, refer to structure Table 3-45. Structure cau_context_parameter_struct

Return value	
-	-

Example:

```

cau_context_parameter_struct context;

cau_key_parameter_struct key;

cau_parameter_struct cau_parameter;

uint32_t keyaddr;

.....

keyaddr = (uint32_t)(cau_parameter->key);

cau_key_struct_para_init(&key);

key.key_1_high = __REV(*(uint32_t*)(keyaddr));

keyaddr += 4U;

key.key_1_low = __REV(*(uint32_t*)(keyaddr));

/* save context before context switching */

cau_context_save(&context, &key);

```

cau_context_restore

The description of cau_context_restore is shown as below:

Table 3-66. Function cau_context_restore

Function name	cau_context_restore
Function prototype	void cau_context_restore(cau_context_parameter_struct *cau_context);
Function descriptions	restore context after context switching
Precondition	-
The called functions	-
Input parameter{in}	
cau_context	the context, refer to structure Table 3-45. Structure cau_context_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

cau_context_parameter_struct context;

.....

```

```
cau_context_save(&context, &key);
```

```
.....
```

```
/* restore context after context switching */
```

```
cau_context_restore (&context);
```

cau_aes_ecb

The description of cau_aes_ecb is shown as below:

Table 3-67. Function cau_aes_ecb

Function name	cau_aes_ecb
Function prototype	ErrStatus cau_aes_ecb(cau_parameter_struct *cau_parameter, uint8_t *output);
Function descriptions	encrypt and decrypt using AES in ECB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;
```

```
uint8_t encrypt_result[TEXT_SIZE];
```

```
ErrStatus status;
```

```
.....
```

```
cau_struct_para_init(&text);
```

```
key_addr = key_select[i];
```

```
key_size = keysize[i];
```

```
text.alg_dir = CAU_ENCRYPT;
```

```
text.key = key_addr;
```

```
text.key_size = key_size;
```

```
text.input = plaintext;
```

```
text.in_length = TEXT_SIZE;
```

```
/* encryption in ECB mode */
```

```
status = cau_aes_ecb(&text, encrypt_result);
```

cau_aes_cbc

The description of cau_aes_cbc is shown as below:

Table 3-68. Function cau_aes_cbc

Function name	cau_aes_cbc
Function prototype	ErrStatus cau_aes_cbc(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using AES in CBC mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;
```

```
uint8_t encrypt_result[TEXT_SIZE];
```

```
ErrStatus status;
```

```
.....
```

```
cau_struct_para_init(&text);
```

```
key_addr = key_select[i];
```

```
key_size = keysize[i];
```

```
text.alg_dir = CAU_ENCRYPT;
```

```
text.key = key_addr;
```

```
text.key_size = key_size;
```

```
text.iv = vectors;
```

```
text.input = plaintext;
```

```
text.in_length = TEXT_SIZE;
```

```
/* encryption in CBC mode */
```

```
status = cau_aes_cbc(&text, encrypt_result);
```

cau_aes_ctr

The description of cau_aes_ctr is shown as below:

Table 3-69. Function cau_aes_ctr

Function name	cau_aes_ctr
Function prototype	ErrStatus cau_aes_ctr(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using AES in CTR mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

cau_struct_para_init(&text);

key_addr = key_select[i];

key_size = keysize[i];

text.alg_dir    = CAU_ENCRYPT;

text.key        = key_addr;

text.key_size   = key_size;

text.iv         = vectors;

text.input      = plaintext;

text.in_length  = TEXT_SIZE;

/* encryption in CTR mode */

status = cau_aes_ctr(&text, encrypt_result);
```

cau_aes_cfb

The description of cau_aes_cfb is shown as below:

Table 3-70. Function cau_aes_cfb

Function name	cau_aes_cfb
Function prototype	ErrStatus cau_aes_cfb(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using AES in CFB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct cau_cfb_parameter;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

cau_struct_para_init(&cau_cfb_parameter);

/* encryption in CFB mode */

cau_cfb_parameter.alg_dir   = CAU_ENCRYPT;

cau_cfb_parameter.key       = (uint8_t *)key_128;

cau_cfb_parameter.key_size  = KEY_SIZE;

cau_cfb_parameter.iv        = (uint8_t *)vectors;

cau_cfb_parameter.iv_size   = IV_SIZE;

cau_cfb_parameter.input     = (uint8_t *)plaintext;

cau_cfb_parameter.in_length = PLAINTEXT_SIZE;

status = cau_aes_cfb(&cau_cfb_parameter, encrypt_result);
```

cau_aes_ofb

The description of cau_aes_ofb is shown as below:

Table 3-71. Function cau_aes_ofb

Function name	cau_aes_ofb
Function prototype	ErrStatus cau_aes_ofb(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using AES in OFB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct cau_ofb_parameter;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

cau_struct_para_init(&cau_ofb_parameter);

/* encryption in OFB mode */

cau_ofb_parameter.alg_dir = CAU_ENCRYPT;

cau_ofb_parameter.key = (uint8_t *)key_128;

cau_ofb_parameter.key_size = KEY_SIZE;

cau_ofb_parameter.iv = (uint8_t *)vectors;

cau_ofb_parameter.iv_size = IV_SIZE;

cau_ofb_parameter.input = (uint8_t *)plaintext;

cau_ofb_parameter.in_length = PLAINTEXT_SIZE;

status = cau_aes_ofb(&cau_ofb_parameter, encrypt_result);
```

cau_aes_gcm

The description of cau_aes_gcm is shown as below:

Table 3-72. Function cau_aes_gcm

Function name	cau_aes_gcm
Function prototype	ErrStatus cau_aes_gcm(cau_parameter_struct *cau_parameter, uint8_t

	*output, uint8_t *tag)
Function descriptions	encrypt and decrypt using AES in GCM mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Output parameter{out}	
tag	pointer to the returned tag buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct cau_gcm_parameter;

uint8_t encrypt_result[TEXT_SIZE];

uint8_t gcm_tag[GCM_TAG_SIZE];

ErrStatus status;

.....

cau_struct_para_init(&cau_gcm_parameter);

/* encryption in GCM mode */

cau_gcm_parameter.alg_dir    = CAU_ENCRYPT;
cau_gcm_parameter.key        = (uint8_t *)key_128;
cau_gcm_parameter.key_size   = KEY_SIZE;
cau_gcm_parameter.iv         = (uint8_t *)vectors;
cau_gcm_parameter.iv_size    = IV_SIZE;
cau_gcm_parameter.input      = (uint8_t *)plaintext;
cau_gcm_parameter.in_length   = PLAINTEXT_SIZE;
cau_gcm_parameter.aad        = (uint8_t *)aadmessage;
cau_gcm_parameter.aad_size    = AAD_SIZE;

status = cau_aes_gcm(&cau_gcm_parameter, encrypt_result, gcm_tag);
```

cau_aes_ccm

The description of cau_aes_ccm is shown as below:

Table 3-73. Function cau_aes_ccm

Function name	cau_aes_ccm
Function prototype	ErrStatus cau_aes_ccm(cau_parameter_struct *cau_parameter, uint8_t *output, uint8_t tag[], uint32_t tag_size, uint8_t aad_buf[])
Function descriptions	encrypt and decrypt using AES in CCM mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Input parameter{in}	
tag_size	tag size (in bytes)
Output parameter{out}	
output	pointer to the returned buffer
Output parameter{out}	
tag	pointer to the returned tag buffer
Output parameter{out}	
aad_buf	pointer to the user buffer used when formatting aad block
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```

cau_parameter_struct cau_ccm_parameter;

uint8_t encrypt_result[TEXT_SIZE];

uint8_t ccm_tag[CCM_TAG_SIZE];

uint8_t aad_buf[AAD_SIZE + 21];

ErrStatus status;

.....

cau_struct_para_init(&cau_ccm_parameter);

/* encryption in CCM mode */

cau_ccm_parameter.alg_dir    = CAU_ENCRYPT;

cau_ccm_parameter.key        = (uint8_t *)ccm_key_128;

cau_ccm_parameter.key_size   = KEY_SIZE;

cau_ccm_parameter.iv         = (uint8_t *)ccm_vectors;

cau_ccm_parameter.iv_size    = CCM_IV_SIZE;

cau_ccm_parameter.input      = (uint8_t *)plaintext;

```

```
cau_ccm_parameter.in_length = PLAINTEXT_SIZE;

cau_ccm_parameter.aad       = (uint8_t *)aadmessage;

cau_ccm_parameter.aad_size  = AAD_SIZE;

status = cau_aes_ccm(&cau_ccm_parameter, encrypt_result, ccm_tag, CCM_TAG_SIZE,
(uint8_t *)aad_buf);
```

cau_tdes_ecb

The description of cau_tdes_ecb is shown as below:

Table 3-74. Function cau_tdes_ecb

Function name	cau_tdes_ecb
Function prototype	ErrStatus cau_tdes_ecb(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using TDES in ECB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

cau_struct_para_init(&text);

text.alg_dir = CAU_ENCRYPT;

text.key     = tdes_key;

text.input   = plaintext;

text.in_length = DATA_SIZE;

/* encryption in ECB mode */

status = cau_tdes_ecb(&text, encrypt_result);
```

cau_tdes_cbc

The description of cau_tdes_cbc is shown as below:

Table 3-75. Function cau_tdes_cbc

Function name	cau_tdes_cbc
Function prototype	ErrStatus cau_tdes_cbc(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using TDES in CBC mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

cau_struct_para_init(&text);

text.alg_dir    = CAU_ENCRYPT;

text.key        = tdes_key;

text.iv         = vectors;

text.input      = plaintext;

text.in_length = DATA_SIZE;

/* encryption in CBC mode */

status = cau_tdes_cbc(&text, encrypt_result);
```

cau_des_ecb

The description of cau_des_ecb is shown as below:

Table 3-76. Function cau_des_ecb

Function name	cau_des_ecb
----------------------	-------------

Function prototype	ErrStatus cau_des_ecb(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using DES in ECB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46. Structure cau_parameter_struct
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

cau_struct_para_init(&text);

text.alg_dir    = CAU_ENCRYPT;

text.key        = des_key;

text.input      = plaintext;

text.in_length  = DATA_SIZE;

/* encryption in ECB mode */

status = cau_des_ecb(&text, encrypt_result);
```

cau_des_cbc

The description of cau_des_cbc is shown as below:

Table 3-77. Function cau_des_cbc

Function name	cau_des_cbc
Function prototype	ErrStatus cau_des_cbc(cau_parameter_struct *cau_parameter, uint8_t *output)
Function descriptions	encrypt and decrypt using DES in CBC mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	the CAU encrypt and decrypt parameters, refer to structure Table 3-46.

	<u>Structure cau_parameter_struct</u>
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

cau_struct_para_init(&text);

text.alg_dir   = CAU_ENCRYPT;

text.key       = des_key;

text.iv        = vectors;

text.input     = plaintext;

text.in_length = DATA_SIZE;

/* encryption in CBC mode */

status = cau_des_cbc(&text, encrypt_result);
```

cau_interrupt_enable

The description of cau_interrupt_enable is shown as below:

Table 3-78. Function cau_interrupt_enable

Function name	cau_interrupt_enable
Function prototype	void cau_interrupt_enable(uint32_t interrupt)
Function descriptions	enable the CAU interrupts
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify the CAU interrupt source to be enabled
CAU_INT_INFIFO	input FIFO interrupt
CAU_INT_OUTFIFO	output FIFO interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable cau interrupt */

cau_interrupt_enable (CAU_INT_INFIFO);
```

cau_interrupt_disable

The description of cau_interrupt_disable is shown as below:

Table 3-79. Function cau_interrupt_disable

Function name	cau_interrupt_disable
Function prototype	void cau_interrupt_disable(uint32_t interrupt)
Function descriptions	disable the CAU interrupts
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify the CAU interrupt source to be disabled
CAU_INT_INFIFO	input FIFO interrupt
CAU_INT_OUTFIFO	output FIFO interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable cau interrupt */

cau_interrupt_disable (CAU_INT_INFIFO);
```

cau_interrupt_flag_get

The description of cau_interrupt_flag_get is shown as below:

Table 3-80. Function cau_interrupt_flag_get

Function name	cau_interrupt_flag_get
Function prototype	FlagStatus cau_interrupt_flag_get(uint32_t interrupt)
Function descriptions	get the interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	CAU interrupt flag
CAU_INT_FLAG_INFIFO	input FIFO interrupt
CAU_INT_FLAG_OUTFIFO	output FIFO interrupt

Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the CAU interrupt flag status */
```

```
FlagStatus status = cau_interrupt_flag_get(CAU_INT_FLAG_INFIFO);
```

cau_flag_get

The description of cau_flag_get is shown as below:

Table 3-81. Function cau_flag_get

Function name	cau_flag_get
Function prototype	FlagStatus cau_flag_get(uint32_t flag)
Function descriptions	get the CAU flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	CAU flag status
CAU_FLAG_INFIFO_EMPTY	input FIFO empty
CAU_FLAG_INFIFO_NO_FULL:	input FIFO is not full
CAU_FLAG_OUTFIFO_NO_EMPTY	output FIFO not empty
CAU_FLAG_OUTFIFO_FULL	output FIFO is full
CAU_FLAG_BUSY	the CAU core is busy
CAU_FLAG_INFIFO	input FIFO flag status
CAU_FLAG_OUTFIFO	output FIFO flag status
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the CAU flag status */
```

```
FlagStatus status;
```

```
status = cau_flag_get (CAU_FLAG_INFIFO_EMPTY);
```


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 3-82. 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 3-83. CRC firmware function

Function name	Function description
crc_deinit	deinit CRC calculation unit
crc_data_register_reset	reset data register(CRC_DATA) to the value of 0xFFFFFFFF
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 3-84. 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 3-85. Function crc_data_register_reset

Function name	crc_data_register_reset
Function prototype	void crc_data_register_reset(void);
Function descriptions	reset data register(CRC_DATA) to the value of 0xFFFFFFFF
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 3-86. 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 3-87. Function `crc_free_data_register_read`

Function name	<code>crc_free_data_register_read</code>
Function prototype	<code>uint8_t crc_free_data_register_read(void);</code>
Function descriptions	read the value of the free data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>uint8_t</code>	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 3-88. Function `crc_free_data_register_write`

Function name	<code>crc_free_data_register_write</code>
Function prototype	<code>void crc_free_data_register_write(uint8_t free_data);</code>
Function descriptions	write data to the free data register
Precondition	-
The called functions	-
Input parameter{in}	
<code>free_data</code>	specified 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 3-89. 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	specified 32-bit data
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit value calculated by CRC (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 3-90. 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 values

Input parameter{in}	
size	size of the array
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit value calculated by CRC (0-0xFFFFFFFF)

Example:

```
/* 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. DBG

The DBG hold unit helps debugger to debug power saving mode. The DBG registers are listed in chapter [3.5.1](#). the DBG firmware functions are introduced in chapter [3.5.2](#).

3.5.1. Descriptions of Peripheral registers

DBG registers are listed in the table shown as below:

Table 3-91. DBG Registers

Registers	Descriptions
DBG_ID	DBG ID code register
DBG_CTL0	DBG control register0
DBG_CTL1	DBG control register1
DBG_CTL2	DBG control register2

3.5.2. Descriptions of Peripheral functions

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

Table 3-92. DBG firmware function

Function name	Function description
dbg_deinit	deinitialize the DBG
dbg_id_get	read DBG_ID code register

Function name	Function description
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
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
dbg_trace_pin_mode_set	set trace pin mode

Enum dbg_periph_enum

Table 3-93. Enum dbg_periph_enum

Member name	Function description
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_TIMER4_HOLD	hold TIMER4 counter when core is halted
DBG_TIMER5_HOLD	hold TIMER5 counter when core is halted
DBG_RTC_HOLD	hold RTC counter when core is halted
DBG_FWDGT_HOLD	hold FWDGT counter when core is halted
DBG_WWDGT_HOLD	hold WWDGT counter when core is halted
DBG_I2C0_HOLD	hold I2C0 smbus timeout when core is halted
DBG_I2C1_HOLD	hold I2C1 smbus timeout when core is halted
DBG_TIMER0_HOLD	hold TIMER0 counter when core is halted
DBG_TIMER15_HOLD	hold TIMER15 counter when core is halted
DBG_TIMER16_HOLD	hold TIMER16 counter when core is halted

dbg_deinit

The description of dbg_deinit is shown as below:

Table 3-94. Function dbg_deinit

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

Example:

```
/* deinitialize the DBG */
```

```
dbg_deinit();
```

dbg_id_get

The description of dbg_id_get is shown as below:

Table 3-95. 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 3-96. 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

<i>TANDBY</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* keep debugger connection during sleep 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 3-97. 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
<i>DBG_LOW_POWER_SLEEP</i>	keep debugger connection during sleep mode
<i>DBG_LOW_POWER_DEEPSLEEP</i>	keep debugger connection during deepsleep mode
<i>DBG_LOW_POWER_STANDBY</i>	keep debugger connection during standby mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* do not keep debugger connection during sleep mode */
```

```
dbg_low_power_disable(DBG_LOW_POWER_SLEEP);
```

dbg_periph_enable

The description of dbg_periph_enable is shown as below:

Table 3-98. 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 3-93. Enum dbg_periph_enum
<i>DBG_FWDGT_HOLD</i>	debug FWDGT kept when core is halted
<i>DBG_WWDGT_HOLD</i>	debug WWDGT kept when core is halted
<i>DBG_I2Cx_HOLD</i>	x=0,1 hold I2Cx smbus when core is halted
<i>DBG_TIMERx_HOLD</i>	x=0,1,2,3,4,5,15,16, hold TIMERx counter when core is halted
<i>DBG_RTC_HOLD</i>	hold RTC counter when core is halted
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* hold TIMERO counter when core is halted */
```

```
dbg_periph_enable(DBG_TIMER0_HOLD);
```

dbg_periph_disable

The description of dbg_periph_disable is shown as below:

Table 3-99. 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 3-93. Enum dbg_periph_enum
<i>DBG_FWDGT_HOLD</i>	debug FWDGT kept when core is halted
<i>DBG_WWDGT_HOLD</i>	debug WWDGT kept when core is halted
<i>DBG_I2Cx_HOLD</i>	x=0,1 hold I2Cx smbus when core is halted
<i>DBG_TIMERx_HOLD</i>	x=0,1,2,3,4,5,15,16, hold TIMERx counter when core is halted
<i>DBG_RTC_HOLD</i>	hold RTC counter when core is halted
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* do not hold TIMERO counter when core is halted */
```

dbg_periph_disable(DBG_TIMER0_HOLD);

dbg_trace_pin_enable

The description of dbg_trace_pin_enable is shown as below:

Table 3-100. 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 3-101. 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();
```

dbg_trace_pin_mode_set

The description of dbg_trace_pin_mode_set is shown as below:

Table 3-102. Function dbg_trace_pin_mode_set

Function name	dbg_trace_pin_mode_set
Function prototype	void dbg_trace_pin_mode_set(uint32_t trace_mode);
Function descriptions	trace pin mode selection
Precondition	-
The called functions	-
Input parameter{in}	
trace_mode	trace pin mode selection
TRACE_MODE_ASYNC	trace pin used for async mode
TRACE_MODE_SYNC_DATASIZE_1	trace pin used for sync mode and data size is 1
TRACE_MODE_SYNC_DATASIZE_2	trace pin used for sync mode and data size is 2
TRACE_MODE_SYNC_DATASIZE_4	trace pin used for sync mode and data size is 4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* trace pin used for async mode */
```

```
dbg_trace_pin_mode_set(TRACE_MODE_ASYNC);
```

3.6. DCI

The DCI is a parallel interface able to capture video or picture from a camera via Digital Camera Interface. The DCI registers are listed in chapter [3.6.1](#). the DCI firmware functions are introduced in chapter [3.6.2](#).

3.6.1. Descriptions of Peripheral registers

DCI registers are listed in the table shown as below:

Table 3-103. DCI Registers

Registers	Descriptions
DCI_CTL	DCI control register
DCI_STAT0	DCI status register 0

Registers	Descriptions
DCI_STAT1	DCI status register 1
DCI_INTEN	DCI interrupt enable register
DCI_INTF	DCI interrupt flag register
DCI_INTC	DCI interrupt clear register
DCI_SC	DCI synchronization codes register
DCI_SCUMSK	DCI synchronization codes unmask register
DCI_CWSPOS	DCI cropping window start position register
DCI_CWSZ	DCI cropping window size register
DCI_DATA	DCI data register

3.6.2. Descriptions of Peripheral functions

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

Table 3-104. DCI firmware function

Function name	Function description
dc_i_deinit	DCI deinit
dc_i_struct_para_init	initialize dc_i_parameter_struct with the default values
dc_i_init	initialize DCI registers
dc_i_enable	enable DCI function
dc_i_disable	disble DCI function
dc_i_capture_enable	enable DCI capture
dc_i_capture_disable	disble DCI capture
dc_i_jpeg_enable	enable DCI jpeg mode
dc_i_jpeg_disable	disble DCI jpeg mode
dc_i_crop_window_enable	enable cropping window function
dc_i_crop_window_disable	disble cropping window function
dc_i_crop_window_config	config DCI cropping window
dc_i_embedded_sync_enable	enable embedded synchronous mode
dc_i_embedded_sync_disable	disble embedded synchronous mode
dc_i_sync_codes_config	config synchronous codes in embedded synchronous mode
dc_i_sync_codes_unmask_config	config synchronous codes unmask in embedded synchronous mode
dc_i_data_read	read DCI data register
dc_i_flag_get	get specified flag
dc_i_interrupt_enable	enable specified DCI interrupt
dc_i_interrupt_disable	disble specified DCI interrupt
dc_i_interrupt_flag_get	get specified interrupt flag
dc_i_interrupt_flag_clear	clear specified interrupt flag

Structure dci_parameter_struct

Table 3-105. Structure dci_parameter_struct

Member name	Function description
capture_mode	DCI capture mode: DCI_CAPTURE_MODE_CONTINUOUS / DCI_CAPTURE_MODE_SNAPSHOT
clock_polarity	clock polarity selection: DCI_CK_POLARITY_FALLING / DCI_CK_POLARITY_RISING
hsync_polarity	horizontal polarity selection: DCI_HSYNC_POLARITY_LOW / DCI_HSYNC_POLARITY_HIGH
vsync_polarity	vertical polarity selection: DCI_VSYNC_POLARITY_LOW / DCI_VSYNC_POLARITY_HIGH
frame_rate	frame capture rate: DCI_FRAME_RATE_ALL / DCI_FRAME_RATE_1_2 / DCI_FRAME_RATE_1_4
interface_format	digital camera interface format: DCI_INTERFACE_FORMAT_8BITS / DCI_INTERFACE_FORMAT_10BITS / DCI_INTERFACE_FORMAT_12BITS / DCI_INTERFACE_FORMAT_14BITS

dci_deinit

The description of dci_deinit is shown as below:

Table 3-106. Function dci_deinit

Function name	dci_deinit
Function prototype	void dci_deinit(void);
Function descriptions	DCI deinit
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DCI deinit */
```

```
dci_deinit();
```

dci_struct_para_init

The description of dci_struct_para_init is shown as below:

Table 3-107. Function dci_struct_para_init

Function name	dci_struct_para_init
Function prototype	void dci_struct_para_init(dci_parameter_struct* init_struct);
Function descriptions	initialize dci_parameter_struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
dci_struct	address of DCI parameter initialization struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize DCI struct parameter */

dci_parameter_struct dci_struct;

dci_struct_para_init(&dci_struct);
```

dci_init

The description of dci_init is shown as below:

Table 3-108. Function dci_init

Function name	dci_init
Function prototype	void dci_init(dci_parameter_struct* dci_struct);
Function descriptions	initialize DCI registers
Precondition	-
The called functions	-
Input parameter{in}	
dci_struct	address of DCI parameter initialization struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize DCI registers */

dci_parameter_struct dci_struct;

dci_struct.capture_mode = DCI_CAPTURE_MODE_CONTINUOUS;

dci_struct.clock_polarity = DCI_CK_POLARITY_RISING;

dci_struct.hsync_polarity = DCI_HSYNC_POLARITY_LOW;
```

```
dc_i_struct.vsync_polarity = DCI_VSYNC_POLARITY_LOW;
```

```
dc_i_struct.frame_rate = DCI_FRAME_RATE_ALL;
```

```
dc_i_struct.interface_format = DCI_INTERFACE_FORMAT_8BITS;
```

```
dc_i_init(&dc_i_struct);
```

dc_i_enable

The description of dc_i_enable is shown as below:

Table 3-109. Function dc_i_enable

Function name	dc_i_enable
Function prototype	void dc_i_enable(void);
Function descriptions	enable DCI function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DCI function */
```

```
dc_i_enable();
```

dc_i_disable

The description of dc_i_disable is shown as below:

Table 3-110. Function dc_i_disable

Function name	dc_i_disable
Function prototype	void dc_i_disable(void);
Function descriptions	disable DCI function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DCI function */
```

```
dc_i_disable();
```

dc_i_capture_enable

The description of dc_i_capture_enable is shown as below:

Table 3-111. Function dc_i_capture_enable

Function name	dc_i_capture_enable
Function prototype	void dc_i_capture_enable(void);
Function descriptions	enable DCI capture
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DCI capture */
```

```
dc_i_capture_enable();
```

dc_i_capture_disable

The description of dc_i_capture_disable is shown as below:

Table 3-112. Function dc_i_capture_disable

Function name	dc_i_capture_disable
Function prototype	void dc_i_capture_disable(void);
Function descriptions	disable DCI capture
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:


```
/* disable DCI capture */
```

```
dc_i_capture_disable();
```

dc_i_jpeg_enable

The description of dc_i_jpeg_enable is shown as below:

Table 3-113. Function dc_i_jpeg_enable

Function name	dc_i_jpeg_enable
Function prototype	void dc_i_jpeg_enable(void);
Function descriptions	enable DCI jpeg mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DCI jpeg mode */
```

```
dc_i_jpeg_enable();
```

dc_i_jpeg_disable

The description of dc_i_jpeg_disable is shown as below:

Table 3-114. Function dc_i_jpeg_disable

Function name	dc_i_jpeg_disable
Function prototype	void dc_i_jpeg_disable(void);
Function descriptions	disable DCI jpeg mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DCI jpeg mode */
```

```
dc_ijpeg_disable();
```

dc_iCrop_window_enable

The description of dc_iCrop_window_enable is shown as below:

Table 3-115. Function dc_iCrop_window_enable

Function name	dc_iCrop_window_enable
Function prototype	void dc_iCrop_window_enable(void);
Function descriptions	enable cropping window function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable cropping window function */
```

```
dc_iCrop_window_enable();
```

dc_iCrop_window_disable

The description of dc_iCrop_window_disable is shown as below:

Table 3-116. Function dc_iCrop_window_disable

Function name	dc_iCrop_window_disable
Function prototype	void dc_iCrop_window_disable(void);
Function descriptions	disable cropping window function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable cropping window function */
```

```
dc_iCrop_window_disable();
```

dc_i_crop_window_config

The description of dc_i_crop_window_config is shown as below:

Table 3-117. Function dc_i_crop_window_config

Function name	dc_i_crop_window_config
Function prototype	void dc_i_crop_window_config(uint16_t start_x, uint16_t start_y, uint16_t size_width, uint16_t size_height);
Function descriptions	config DCI cropping window
Precondition	-
The called functions	-
Input parameter{in}	
start_x	window horizontal start position
Input parameter{in}	
start_y	window vertical start position
Input parameter{in}	
size_width	window horizontal size
Input parameter{in}	
size_height	window vertical size
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config DCI cropping window */
```

```
dc_i_crop_window_config (0x800, 0x600, 0x100, 0x100);
```

dc_i_embedded_sync_enable

The description of dc_i_embedded_sync_enable is shown as below:

Table 3-118. Function dc_i_embedded_sync_enable

Function name	dc_i_embedded_sync_enable
Function prototype	void dc_i_embedded_sync_enable(void);
Function descriptions	enable embedded synchronous mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable embedded synchronous mode */
```

```
dc_i_embedded_sync_enable();
```

dc_i_embedded_sync_disable

The description of dc_i_embedded_sync_disable is shown as below:

Table 3-119. Function dc_i_embedded_sync_disable

Function name	dc_i_embedded_sync_disable
Function prototype	void dc_i_embedded_sync_disable(void);
Function descriptions	disable embedded synchronous mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable embedded synchronous mode */
```

```
dc_i_embedded_sync_disable();
```

dc_i_sync_codes_config

The description of dc_i_sync_codes_config is shown as below:

Table 3-120. Function dc_i_sync_codes_config

Function name	dc_i_sync_codes_config
Function prototype	void dc_i_sync_codes_config(uint8_t frame_start, uint8_t line_start, uint8_t line_end, uint8_t frame_end);
Function descriptions	config synchronous codes in embedded synchronous mode
Precondition	-
The called functions	-
Input parameter{in}	
frame_start	frame start code in embedded synchronous mode
Input parameter{in}	
line_start	line start code in embedded synchronous mode
Input parameter{in}	
line_end	line end code in embedded synchronous mode
Input parameter{in}	

frame_end	frame end code in embedded synchronous mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config synchronous codes in embedded synchronous mode */
dci_sync_codes_config(0x10, 0x10, 0x20, 0x20);
```

dci_sync_codes_unmask_config

The description of dci_sync_codes_unmask_config is shown as below:

Table 3-121. Function dci_sync_codes_unmask_config

Function name	dci_sync_codes_unmask_config
Function prototype	void dci_sync_codes_unmask_config(uint8_t frame_start, uint8_t line_start, uint8_t line_end, uint8_t frame_end);
Function descriptions	config synchronous codes unmask in embedded synchronous mode
Precondition	-
The called functions	-
Input parameter{in}	
frame_start	frame start code in embedded synchronous mode
Input parameter{in}	
line_start	line start code in embedded synchronous mode
Input parameter{in}	
line_end	line end code in embedded synchronous mode
Input parameter{in}	
frame_end	frame end code in embedded synchronous mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config synchronous codes unmask in embedded synchronous mode */
dci_sync_codes_unmask_config(0x10, 0x10, 0x20, 0x20);
```

dci_data_read

The description of dci_data_read is shown as below:

Table 3-122. Function dci_data_read

Function name	dci_data_read
----------------------	---------------

Function prototype	uint32_t dci_data_read(void);
Function descriptions	read DCI data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	0x00 – 0xFFFFFFFF

Example:

```
/* read DCI data register */
uint32_t data = dci_data_read();
```

dc_i_flag_get

The description of dc_i_flag_get is shown as below:

Table 3-123. Function dc_i_flag_get

Function name	dc_i_flag_get
Function prototype	FlagStatus dc_i_flag_get(uint32_t flag);
Function descriptions	get specified flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	DCI flag
DCI_FLAG_HS	HS line status
DCI_FLAG_VS	VS line status
DCI_FLAG_FV	FIFO valid
DCI_FLAG_EF	end of frame flag
DCI_FLAG_OVR	FIFO overrun flag
DCI_FLAG_ESE	embedded synchronous error flag
DCI_FLAG_VSYNC	vsync flag
DCI_FLAG_EL	end of line flag
Output parameter{out}	
-	-
Return value	
FlagStatus	FlagStatus: SET or RESET

Example:

```
/* get specified flag */
FlagStatus status = dc_i_flag_get(DCI_FLAG_HS);
```

dc_i_interrupt_enable

The description of dc_i_interrupt_enable is shown as below:

Table 3-124. Function dc_i_interrupt_enable

Function name	dc_i_interrupt_enable
Function prototype	void dc_i_interrupt_enable(uint32_t interrupt);
Function descriptions	enable specified DCI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	DCI interrupt
<i>DCI_INT_EF</i>	end of frame interrupt
<i>DCI_INT_OVR</i>	FIFO overrun interrupt
<i>DCI_INT_ESE</i>	embedded synchronous error interrupt
<i>DCI_INT_VSYNC</i>	vsync interrupt
<i>DCI_INT_EL</i>	end of line interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable specified DCI interrupt */
dc_i_interrupt_enable(DCI_INT_EF);
```

dc_i_interrupt_disable

The description of dc_i_interrupt_disable is shown as below:

Table 3-125. Function dc_i_interrupt_disable

Function name	dc_i_interrupt_disable
Function prototype	void dc_i_interrupt_disable(uint32_t interrupt);
Function descriptions	disable specified DCI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	DCI interrupt
<i>DCI_INT_EF</i>	end of frame interrupt
<i>DCI_INT_OVR</i>	FIFO overrun interrupt
<i>DCI_INT_ESE</i>	embedded synchronous error interrupt
<i>DCI_INT_VSYNC</i>	vsync interrupt
<i>DCI_INT_EL</i>	end of line interrupt
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable specified DCI interrupt */
```

```
dc_iinterrupt_disable(DCI_INT_EF);
```

dc_iinterrupt_flag_get

The description of dc_iinterrupt_flag_get is shown as below:

Table 3-126. Function dc_iinterrupt_flag_get

Function name	dc_iinterrupt_flag_get
Function prototype	FlagStatus dc_iinterrupt_flag_get(uint32_t interrupt);
Function descriptions	get specified interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	DCI interrupt
DCI_INT_FLAG_EF	end of frame interrupt flag
DCI_INT_FLAG_OVR	FIFO overrun interrupt flag
DCI_INT_FLAG_ESE	embedded synchronous error interrupt flag
DCI_INT_FLAG_VSYN C	vsync interrupt flag
DCI_INT_FLAG_EL	end of line interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	FlagStatus: SET or RESET

Example:

```
/* get specified interrupt flag */
```

```
FlagStatus status = dc_iinterrupt_flag_get(DCI_INT_FLAG_EF);
```

dc_iinterrupt_flag_clear

The description of dc_iinterrupt_flag_clear is shown as below:

Table 3-127. Function dc_iinterrupt_flag_clear

Function name	dc_iinterrupt_flag_clear
Function prototype	void dc_iinterrupt_flag_clear(uint32_t interrupt);
Function descriptions	clear specified interrupt flag
Precondition	-

The called functions	-
Input parameter{in}	
interrupt	DCI interrupt
<i>DCI_INT_FLAG_EF</i>	end of frame interrupt flag
<i>DCI_INT_FLAG_OVR</i>	FIFO overrun interrupt flag
<i>DCI_INT_FLAG_ESE</i>	embedded synchronous error interrupt flag
<i>DCI_INT_FLAG_VSYN</i> <i>C</i>	vsync interrupt flag
<i>DCI_INT_FLAG_EL</i>	end of line interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear specified interrupt flag */
dci_interrupt_flag_clear (DCI_INT_FLAG_EF);
```

3.7. 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 [3.7.1](#), the DMA firmware functions are introduced in chapter [3.7.2](#).

3.7.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

Table 3-128. DMA Registers

Registers	Descriptions
DMA_INTF0	Interrupt flag register0
DMA_INTF1	Interrupt flag register1
DMA_INTC0	Interrupt flag clear register0
DMA_INTC1	Interrupt flag clear register1
DMA_CHxCTL (x=0..7)	Channel x control register
DMA_CHxCNT (x=0..7)	Channel x counter register
DMA_CHxPADDR (x=0..7)	Channel x peripheral base address register

Registers	Descriptions
DMA_CHxM0ADDR (x=0..7)	Channel x memory 0 base address register
DMA_CHxM1ADDR (x=0..7)	Channel x memory 1 base address register
DMA_CHxFCTL	Channel x FIFO control register
DMA_SSTAT	Security status register
DMA_SSC	Security status clear register
DMA_CHxSCTL	Channel x security control register

3.7.2. Descriptions of Peripheral functions

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

Table 3-129. DMA firmware function

Function name	Function description
dma_deinit	deinitialize DMA a channel registers
dma_single_struct_para_struct_init	initialize the DMA single data mode parameters struct with the default values
dma_multi_struct_para_struct_init	initialize the DMA multi data mode parameters struct with the default values
dma_single_data_mode_init	DMA single data mode initialize
dma_multi_data_mode_init	DMA multi data mode initialize
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_burst_beats_config	configure transfer burst beats of memory
dma_periph_burst_beats_config	configure transfer burst beats of peripheral
dma_memory_width_config	configure transfer data size of memory
dma_periph_width_config	configure transfer data size of peripheral
dma_memory_address_generation_config	configure next address increasement algorithm of memory
dma_peripheral_address_generation_config	configure next address increasement algorithm of peripheral
dma_circulation_enable	enable DMA circulation mode
dma_circulation_disable	disable DMA circulation mode
dma_channel_enable	enable DMA channel
dma_channel_disable	disable DMA channel
dma_transfer_direction_config	configure the direction of data transfer on the channel

Function name	Function description
<code>dma_switch_buffer_mode_config</code>	configure DMA switch buffer mode
<code>dma_using_memory_get</code>	get DMA using memory
<code>dma_channel_subperipheral_select</code>	select DMA channel peripheral
<code>dma_flow_controller_config</code>	configure DMA flow controller
<code>dma_switch_buffer_mode_enable</code>	enable DMA switch buffer mode
<code>dma_switch_buffer_mode_disable</code>	disable DMA switch buffer mode
<code>dma_fifo_status_get</code>	get DMA FIFO status
<code>dma_flag_get</code>	check the designated flag of a DMA channel is set or not
<code>dma_flag_clear</code>	clear the designated flag of a DMA channel
<code>dma_interrupt_flag_get</code>	check the designated interrupt flag of a DMA channel is set or not
<code>dma_interrupt_flag_clear</code>	clear the designated interrupt flag of a DMA channel
<code>dma_interrupt_enable</code>	enable DMA interrupt
<code>dma_interrupt_disable</code>	disable DMA interrupt
<code>dma_security_config</code>	configure secure attribution of DMA channel
<code>dma_priv_config</code>	configure privileged mode of DMA channel
<code>dma_illegal_access_flag_get</code>	check DMA channel illegal flag is set or not
<code>dma_illegal_access_flag_clear</code>	clear DMA channel illegal flag

Enum `dma_channel_enum`

Table 3-130. Enum `dma_channel_enum`

Member name	Function description
<code>DMA_CH0</code>	DMA Channel0
<code>DMA_CH1</code>	DMA Channel1
<code>DMA_CH2</code>	DMA Channel2
<code>DMA_CH3</code>	DMA Channel3
<code>DMA_CH4</code>	DMA Channel4
<code>DMA_CH5</code>	DMA Channel5
<code>DMA_CH6</code>	DMA Channel6
<code>DMA_CH7</code>	DMA Channel7

Enum `dma_subperipheral_enum`

Table 3-131. Enum `dma_subperipheral_enum`

Member name	Function description
<code>DMA_SUBPERI0</code>	DMA Peripheral 0
<code>DMA_SUBPERI1</code>	DMA Peripheral 1
<code>DMA_SUBPERI2</code>	DMA Peripheral 2
<code>DMA_SUBPERI3</code>	DMA Peripheral 3
<code>DMA_SUBPERI4</code>	DMA Peripheral 4
<code>DMA_SUBPERI5</code>	DMA Peripheral 5

DMA_SUBPERI6	DMA Peripheral 6
DMA_SUBPERI7	DMA Peripheral 7

Structure dma_multi_data_parameter_struct

Table 3-132. Structure dma_multi_data_parameter_struct

Member name	Function description
periph_addr	peripheral base address
periph_width	transfer data size of peripheral
periph_inc	peripheral increasing mode
memory0_addr	memory 0 base address
memory_width	transfer data size of memory
memory_inc	memory increasing mode
memory_burst_width	multi data mode enable
periph_burst_width	multi data mode enable
critical_value	FIFO critical
circular_mode	DMA circular mode
direction	channel data transfer direction
number	channel transfer number
priority	channel priority level

Structure dma_single_data_parameter_struct

Table 3-133. Structure dma_single_data_parameter_struct

Member name	Function description
periph_addr	peripheral base address
periph_inc	peripheral increasing mode
memory0_addr	memory 0 base address
memory_inc	memory increasing mode
periph_memory_width	transfer data size of peripheral
circular_mode	DMA circular mode
direction	channel data transfer direction
number	channel transfer number
priority	channel priority level

dma_deinit

The description of dma_deinit is shown as below:

Table 3-134. 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
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	specify which DMA channel is deinitialized, refer to Table 3-130. Enum dma_channel_enum
<i>DMA_CHx(x=0..7)</i>	DMA channel selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

dma_single_data_para_struct_init

The description of dma_single_data_para_struct_init is shown as below:

Table 3-135. Function dma_single_data_para_struct_init

Function name	dma_single_data_para_struct_init
Function prototype	void dma_single_data_para_struct_init(dma_single_data_parameter_struct* init_struct);
Function descriptions	initialize the DMA single data mode parameters struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	the address of structure for initialization, the structure members can refer to Table 3-133. Structure dma_single_data_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters struct of DMA with single data mode*/
dma_single_data_parameter_struct dma_init_struct;
dma_single_data_para_struct_init(&dma_init_struct);
```

dma_multi_data_para_struct_init

The description of dma_multi_data_para_struct_init is shown as below:

Table 3-136. Function dma_multi_data_para_struct_init

Function name	dma_multi_data_para_struct_init
Function prototype	void dma_multi_data_para_struct_init(dma_multi_data_parameter_struct* init_struct);
Function descriptions	initialize the DMA multi data mode parameters struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	the address of structure for initialization, the structure members can refer to Table 3-132. Structure dma_multi_data_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters struct of DMA with multi data mode*/
dma_multi_data_parameter_struct dma_init_struct;
dma_multi_data_para_struct_init (&dma_init_struct);
```

dma_single_data_mode_init

The description of dma_single_data_mode_init is shown as below:

Table 3-137. Function dma_single_data_mode_init

Function name	dma_single_data_mode_init
Function prototype	void dma_single_data_mode_init(uint32_t dma_periph, dma_channel_enum channelx, dma_single_data_parameter_struct* init_struct);
Function descriptions	initialize DMA single data mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	specify which DMA channel is deinitialized, refer to Table 3-130. Enum dma_channel_enum
<i>DMA_CHx(x=0..7)</i>	DMA channel selection
Input parameter{in}	
init_struct	the address of structure for initialization, the structure members can refer to Table 3-133. Structure dma_single_data_parameter_struct

Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* DMA1 channel0 single data mode initialize */
dma_single_data_parameter_struct dma_init_struct;
dma_deinit(DMA1, DMA_CH0);
dma_single_data_para_struct_init(&dma_init_struct);

dma_init_struct.direction = DMA_MEMORY_TO_MEMORY;
dma_init_struct.memory0_addr = (uint32_t)g_destbuf;
dma_init_struct.memory_inc = DMA_MEMORY_INCREASE_ENABLE;
dma_init_struct.periph_memory_width = DMA_MEMORY_WIDTH_32BIT;
dma_init_struct.number = TRANSFER_NUM;
dma_init_struct.circular_mode = DMA_CIRCULAR_MODE_DISABLE;
dma_init_struct.periph_addr = (uint32_t)FLASH_WRITE_START_ADDR;
dma_init_struct.periph_inc = DMA_PERIPH_INCREASE_ENABLE;
dma_init_struct.priority = DMA_PRIORITY_ULTRA_HIGH;
dma_single_data_mode_init(DMA1, DMA_CH0, &dma_init_struct);

```

dma_multi_data_mode_init

The description of dma_multi_data_mode_init is shown as below:

Table 3-138. Function dma_multi_data_mode_init

Function name	dma_multi_data_mode_init
Function prototype	void dma_multi_data_mode_init(uint32_t dma_periph, dma_channel_enum channelx, dma_multi_data_parameter_struct* init_struct);
Function descriptions	initialize DMA multi data mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	specify which DMA channel is deinitialized, refer to Table 3-130. Enum dma_channel_enum
<i>DMA_CHx(x=0..7)</i>	DMA channel selection
Input parameter{in}	
init_struct	the address of structure for initialization, the structure members can refer to Table 3-132. Structure dma_multi_data_parameter_struct
Output parameter{out}	

-	-
Return value	
-	-

Example:

```

/* DMA0 channel0 multi data mode initialize */
dma_multi_data_parameter_struct dma_init_parameter;
dma_multi_data_para_struct_init(&dma_init_parameter);
dma_deinit(DMA1, DMA_CH0);

dma_init_parameter.periph_addr = (uint32_t)source;
dma_init_parameter.periph_width = DMA_PERIPH_WIDTH_16BIT;
dma_init_parameter.periph_inc = DMA_PERIPH_INCREASE_ENABLE;
dma_init_parameter.memory0_addr = (uint32_t)destination;
dma_init_parameter.memory_width = DMA_MEMORY_WIDTH_16BIT;
dma_init_parameter.memory_inc = DMA_MEMORY_INCREASE_ENABLE;
dma_init_parameter.memory_burst_width = DMA_MEMORY_BURST_4_BEAT;
dma_init_parameter.periph_burst_width = DMA_PERIPH_BURST_4_BEAT;
dma_init_parameter.critical_value = DMA_FIFO_2_WORD;
dma_init_parameter.circular_mode = DMA_CIRCULAR_MODE_DISABLE;
dma_init_parameter.direction = DMA_MEMORY_TO_MEMORY;
dma_init_parameter.number = BUFFER_SIZE;
dma_init_parameter.priority = DMA_PRIORITY_ULTRA_HIGH;
dma_multi_data_mode_init(DMA1,DMA_CH0, &dma_init_parameter);

```

dma_periph_address_config

The description of dma_periph_address_config is shown as below:

Table 3-139. 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
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
address	peripheral base address

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 peripheral address */

#define USART_DATA_ADDR (USART0 + 0x00000024U)
dma_periph_address_config (DMA0, DMA_CH0, USART_DATA_ADDR);
```

dma_memory_address_config

The description of dma_memory_address_config is shown as below:

Table 3-140. 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, uint8_t memory_flag, uint32_t address);
Function descriptions	set DMA memory0 base address
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
address	memory0 base address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 memory0 address */

uint32_t arry[10] = {0};
dma_memory_address_config (DMA0, DMA_CH0, arry);
```

dma_transfer_number_config

The description of dma_transfer_number_config is shown as below:

Table 3-141. 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(x=0...7)	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
number	data transfer number (0x00000000 – 0x0000FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 transfer number */
#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 3-142. 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(x=0...7)	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Output parameter{out}	

-	-
Return value	
uint32_t	0x00000000 – 0x0000FFFF

Example:

```
/* get channel0 memory0 address */
```

```
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 3-143. 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
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
priority	priority Level of this channel
<i>DMA_PRIORITY_LOW</i>	low priority
<i>DMA_PRIORITY_MEDIUM</i>	medium priority
<i>DMA_PRIORITY_HIGH</i>	high priority
<i>DMA_PRIORITY_ULTRA_HIGH</i>	ultra high priority
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 priority */
```

```
dma_priority_config(DMA0, DMA_CH0, DMA_PRIORITY_ULTRA_HIGH);
```

dma_memory_burst_beats_config

The description of dma_memory_burst_beats_config is shown as below:

Table 3-144. Function dma_memory_burst_beats_config

Function name	dma_memory_burst_beats_config
Function prototype	void dma_memory_burst_beats_config (uint32_t dma_periph, dma_channel_enum channelx, uint32_t mbeat);
Function descriptions	configure transfer burst beats of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
mbeat	transfer burst beats
<i>DMA_MEMORY_BURST_SINGLE</i>	memory transfer single burst
<i>DMA_MEMORY_BURST_4_BEAT</i>	memory transfer 4-beat burst
<i>DMA_MEMORY_BURST_8_BEAT</i>	memory transfer 8-beat burst
<i>DMA_MEMORY_BURST_16_BEAT</i>	memory transfer 16-beat burst
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 transfer burst beats of memory */
dma_memory_burst_beats_config(DMA0, DMA_CH0, DMA_MEMORY_BURST_8_BEAT);
```

dma_periph_burst_beats_config

The description of dma_periph_burst_beats_config is shown as below:

Table 3-145. Function dma_periph_burst_beats_config

Function name	dma_periph_burst_beats_config
Function prototype	void dma_periph_burst_beats_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t pbeat);

Function descriptions	configure transfer burst beats of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
pbeat	transfer burst beats
<i>DMA_PERIPH_BURST_SINGLE</i>	peripheral transfer single burst
<i>DMA_PERIPH_BURST_4_BEAT</i>	peripheral transfer 4-beat burst
<i>DMA_PERIPH_BURST_8_BEAT</i>	peripheral transfer 8-beat burst
<i>DMA_PERIPH_BURST_16_BEAT</i>	peripheral transfer 16-beat burst
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 transfer burst beats of peripheral */
dma_periph_burst_beats_config(DMA0, DMA_CH0, DMA_PERIPH_BURST_8_BEAT);
```

dma_memory_width_config

The description of dma_memory_width_config is shown as below:

Table 3-146. 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 msize);
Function descriptions	configure transfer data size of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	

channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
msize	transfer data width of memory
<i>DMA_MEMORY_WIDT H_8BIT</i>	transfer data width of memory is 8-bit
<i>DMA_MEMORY_WIDT H_16BIT</i>	transfer data width of memory is 16-bit
<i>DMA_MEMORY_WIDT H_32BIT</i>	transfer data width of memory is 32-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 transfer memory width */
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 3-147. 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 psize);
Function descriptions	configure transfer data size of peripheral
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
psize	transfer data width of peripheral
<i>DMA_PERIPHERAL_W IDTH_8BIT</i>	transfer data width of peripheral is 8-bit
<i>DMA_PERIPHERAL_W IDTH_16BIT</i>	transfer data width of peripheral is 16-bit
<i>DMA_PERIPHERAL_W</i>	transfer data width of peripheral is 32-bit

<i>IDTH_32BIT</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 transfer peripheral width */
```

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

dma_memory_address_generation_config

The description of dma_memory_address_generation_config is shown as below:

Table 3-148. Function dma_memory_address_generation_config

Function name	dma_memory_address_generation_config
Function prototype	void dma_memory_address_generation_config(uint32_t dma_periph, dma_channel_enum channelx, uint8_t generation_algorithm);
Function descriptions	configure memory address generation algorithm
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
generation_algorithm	memory address generation algorithm
<i>DMA_MEMORY_INCR EASE_ENABLE</i>	next address of memory is increasing address mode
<i>DMA_MEMORY_INCR EASE_DISABLE</i>	next address of memory is fixed address mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 memory address generation algorithm as memory increase*/
```

```
dma_memory_address_generation_config(DMA0, DMA_CH0, DMA_MEMORY_INCREASE_ENABLE);
```

dma_peripheral_address_generation_config

The description of dma_peripheral_address_generation_config is shown as below:

Table 3-149. Function dma_peripheral_address_generation_config

Function name	dma_peripheral_address_generation_config
Function prototype	void dma_peripheral_address_generation_config(uint32_t dma_periph, dma_channel_enum channelx, uint8_t generation_algorithm);
Function descriptions	configure peripheral address generation algorithm
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
generation_algorithm	transfer data width of peripheral
<i>DMA_PERIPH_INCREASE_ENABLE</i>	next address of peripheral is increasing address mode
<i>DMA_PERIPH_INCREASE_DISABLE</i>	next address of peripheral is fixed address mode
<i>DMA_PERIPH_INCREASE_FIX</i>	increasing steps of peripheral address is fixed
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 peripheral address generation algorithm as peripheral increase*/
dma_peripheral_address_generation_config(DMA0, DMA_CH0, DMA_PERIPH_INCREASE_ENABLE);
```

dma_circulation_enable

The description of dma_circulation_enable is shown as below:

Table 3-150. 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
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
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 3-151. 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
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

dma_channel_enable

The description of dma_channel_enable is shown as below:

Table 3-152. 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
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
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 3-153. 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
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Output parameter{out}	

-	-
Return value	
-	-

Example:

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

dma_transfer_direction_config

The description of dma_transfer_direction_config is shown as below:

Table 3-154. 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
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
direction	specify the direction of data transfer
<i>DMA_PERIPHERAL_TO_MEMORY</i>	read from peripheral and write to memory
<i>DMA_MEMORY_TO_PERIPHERAL</i>	read from memory and write to peripheral
<i>DMA_MEMORY_TO_MEMORY</i>	read from memory and write to memory
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 transfer direction as peripheral to memory mode*/
dma_transfer_direction_config(DMA0, DMA_CH0, DMA_PERIPHERAL_TO_MEMORY);
```

dma_switch_buffer_mode_config

The description of dma_switch_buffer_mode_config is shown as below:

Table 3-155. Function dma_switch_buffer_mode_config

Function name	dma_switch_buffer_mode_config
Function prototype	void dma_switch_buffer_mode_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t memory1_addr, uint32_t memory_select);
Function descriptions	configure DMA switch buffer mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
memory1_addr	memory1 base address
Input parameter{in}	
memory_select	select memory transfer area
<i>DMA_MEMORY_0</i>	select memory0 as memory transfer area
<i>DMA_MEMORY_1</i>	select memory1 as memory transfer area
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 switch buffer mode*/
```

```
uint32_t mem[32] = {0};
```

```
dma_switch_buffer_mode_config(DMA0, DMA_CH0, mem, DMA_MEMORY_0);
```

dma_using_memory_get

The description of dma_using_memory_get is shown as below:

Table 3-156. Function dma_using_memory_get

Function name	dma_using_memory_get
Function prototype	uint32_t dma_using_memory_get(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	configure DMA switch buffer mode

Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
memory1_addr	memory1 base address
Output parameter{out}	
-	-
Return value	
uint32_t	the memory buffer area currently used
<i>DMA_MEMORY_0</i>	memory0 is selected as memory transfer area
<i>DMA_MEMORY_1</i>	memory1 is selected as memory transfer area

Example:

```
/* get DMA0 channel0 transfer buffer currently used */
uint32_t buf_addr = DMA_MEMORY_0;
buf_addr = dma_using_memory_get(DMA0, DMA_CH0);
```

dma_channel_subperipheral_select

The description of dma_channel_subperipheral_select is shown as below:

Table 3-157. Function dma_channel_subperipheral_select

Function name	dma_channel_subperipheral_select
Function prototype	void dma_channel_subperipheral_select(uint32_t dma_periph, dma_channel_enum channelx, dma_subperipheral_enum sub_periph);
Function descriptions	DMA channel peripheral select
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
sub_periph	specify DMA channel peripheral
<i>DMA_SUBPERIx(x=0..7)</i>	DMA Peripheral x(x = 0...7)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 peripheral */
```

```
dma_channel_subperipheral_select(DMA0, DMA_CH0, DMA_SUBPERI1);
```

dma_flow_controller_config

The description of dma_flow_controller_config is shown as below:

Table 3-158. Function dma_flow_controller_config

Function name	dma_flow_controller_config
Function prototype	void dma_flow_controller_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t controller);
Function descriptions	configure DMA flow controller
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
controller	specify DMA flow controller
<i>DMA_FLOW_CONTROLLER_DMA</i>	DMA is the flow controller
<i>DMA_FLOW_CONTROLLER_PERI</i>	peripheral is the flow controller
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 flow controller */
```

```
dma_flow_controller_config(DMA0, DMA_CH0, DMA_FLOW_CONTROLLER_DMA);
```

dma_switch_buffer_mode_enable

The description of dma_switch_buffer_mode_enable is shown as below:

Table 3-159. Function dma_switch_buffer_mode_enable

Function name	dma_switch_buffer_mode_enable
Function prototype	void dma_switch_buffer_mode_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA switch buffer mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 channel0 switch buffer mode */
```

```
dma_switch_buffer_mode_enable (DMA0, DMA_CH0);
```

dma_switch_buffer_mode_disable

The description of dma_switch_buffer_mode_disable is shown as below:

Table 3-160. Function dma_switch_buffer_mode_disable

Function name	dma_switch_buffer_mode_disable
Function prototype	void dma_switch_buffer_mode_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA switch buffer mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable DMA0 channel0 switch buffer mode */
```

```
dma_switch_buffer_mode_disable (DMA0, DMA_CH0);
```

dma_fifo_status_get

The description of dma_fifo_status_get is shown as below:

Table 3-161. Function dma_fifo_status_get

Function name	dma_fifo_status_get
Function prototype	uint32_t dma_fifo_status_get(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	get DMA FIFO status
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
uint32_t	the using memory in FIFO

Example:

```
/* get DMA0 channel0 FIFO status */
```

```
uint32_t fifo_state = 0;
```

```
fifo_state = dma_fifo_status_get(DMA0, DMA_CH0);
```

dma_flag_get

The description of dma_flag_get is shown as below:

Table 3-162. 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 the designated flag of a DMA channel is set or not
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
flag	specify get which flag
<i>DMA_FLAG_FEE</i>	FIFO error and exception flag
<i>DMA_FLAG_SDE</i>	single data mode exception flag
<i>DMA_FLAG_TAE</i>	transfer access error flag
<i>DMA_FLAG_HTF</i>	half transfer finish flag
<i>DMA_FLAG_FTF</i>	full transfer finish flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check DMA0 channel0 full transfer finish flag set or not */
```

```
FlagStatus state = RESET;
```

```
state = dma_flag_get(DMA0, DMA_CH0, DMA_FLAG_FTF);
```

dma_flag_clear

The description of dma_flag_clear is shown as below:

Table 3-163. 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 designated flag of a DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum

Input parameter{in}	
flag	specify get which flag
<i>DMA_FLAG_FEE</i>	FIFO error and exception flag
<i>DMA_FLAG_SDE</i>	single data mode exception flag
<i>DMA_FLAG_TAE</i>	transfer access error flag
<i>DMA_FLAG_HTF</i>	half transfer finish flag
<i>DMA_FLAG_FTF</i>	full transfer finish flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMA0 channel0 full transfer finish flag */
```

```
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 3-164. 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 int_flag);
Function descriptions	check the designated interrupt flag of a DMA channel is set or not
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
int_flag	specify get which interrupt flag
<i>DMA_INT_FLAG_FEE</i>	FIFO error and exception interrupt flag
<i>DMA_INT_FLAG_SDE</i>	single data mode exception interrupt flag
<i>DMA_INT_FLAG_TAE</i>	transfer access error interrupt flag
<i>DMA_INT_FLAG_HTF</i>	half transfer finish interrupt flag
<i>DMA_INT_FLAG_FTF</i>	full transfer finish interrupt flag
Output parameter{out}	
-	-
Return value	

FlagStatus	SET or RESET
-------------------	--------------

Example:

```
/* check DMA0 channel0 full transfer finish interrupt flag set or not */
```

```
FlagStatus state = RESET;
```

```
state = dma_interrupt_flag_get (DMA0, DMA_CH0, DMA_INT_FLAG_FTF);
```

dma_interrupt_flag_clear

The description of dma_interrupt_flag_clear is shown as below:

Table 3-165. 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 int_flag);
Function descriptions	clear the designated interrupt flag of a DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
int_flag	specify get which interrupt flag
<i>DMA_INT_FLAG_FEE</i>	FIFO error and exception interrupt flag
<i>DMA_INT_FLAG_SDE</i>	single data mode exception interrupt flag
<i>DMA_INT_FLAG_TAE</i>	transfer access error interrupt flag
<i>DMA_INT_FLAG_HTF</i>	half transfer finish interrupt flag
<i>DMA_INT_FLAG_FTF</i>	full transfer finish interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMA0 channel0 full transfer finish interrupt flag */
```

```
dma_interrupt_flag_clear(DMA0, DMA_CH0, DMA_INT_FLAG_FTF);
```

dma_interrupt_enable

The description of dma_interrupt_enable is shown as below:

Table 3-166. 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
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
source	DMA interrupt source
<i>DMA_INT_FEE</i>	FIFO exception interrupt enable
<i>DMA_INT_SDE</i>	single data mode exception interrupt enable
<i>DMA_INT_TAE</i>	transfer access error interrupt enable
<i>DMA_INT_HTF</i>	half transfer finish interrupt enable
<i>DMA_INT_FTF</i>	full transfer finish interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 channel0 full transfer finish interrupt */
```

```
dma_interrupt_enable(DMA0, DMA_CH0, DMA_INT_FTF);
```

dma_interrupt_disable

The description of dma_interrupt_disable is shown as below:

Table 3-167. 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
<i>DMAx(x=0, 1)</i>	DMA peripheral selection

Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
source	DMA interrupt source
<i>DMA_INT_SDE</i>	single data mode exception interrupt enable
<i>DMA_INT_TAE</i>	transfer access error interrupt enable
<i>DMA_INT_HTF</i>	half transfer finish interrupt enable
<i>DMA_INT_FTF</i>	full transfer finish interrupt enable
<i>DMA_INT_FEE</i>	FIFO exception interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA0 channel0 full transfer finish interrupt */
dma_interrupt_disable(DMA0, DMA_CH0, DMA_INT_FTF);
```

dma_security_config

The description of dma_security_config is shown as below:

Table 3-168. Function dma_security_config

Function name	dma_security_config
Function prototype	void dma_security_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t security);
Function descriptions	configure security attribution of DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
security	security attribution of this channel
<i>DMA_CHANNEL_SECURE</i>	secure mode
<i>DMA_CHANNEL_NON_SECURE</i>	non-secure mode
<i>DMA_CHANNEL_SOURCE</i>	the secure DMA transfer from the source

<i>RCE_SECURE</i>	
<i>DMA_CHANNEL_SOURCE_NONSECURE</i>	the non-secure DMA transfer from the source
<i>DMA_CHANNEL_DESTINATION_SECURE</i>	the secure DMA transfer to the destination
<i>DMA_CHANNEL_DESTINATION_NONSECURE</i>	the non-secure DMA transfer to the destination
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 security attribution */
dma_security_config (DMA0, DMA_CH0, DMA_CHANNEL_SECURE);
```

dma_priv_config

The description of dma_priv_config is shown as below:

Table 3-169. Function dma_priv_config

Function name	dma_priv_config
Function prototype	void dma_priv_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t priv);
Function descriptions	configure privileged mode of DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Input parameter{in}	
priv	privileged mode of this channel
<i>DMA_CHANNEL_PRIV</i>	privileged mode
<i>DMA_CHANNEL_NONPRIV</i>	non-privileged mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel0 privileged mode */
dma_priv_config(DMA0, DMA_CH0, DMA_CHANNEL_PRIV);
```

dma_illegal_access_flag_get

The description of dma_illegal_access_flag_get is shown as below:

Table 3-170. Function dma_illegal_access_flag_get

Function name	dma_illegal_access_flag_get
Function prototype	FlagStatus dma_illegal_access_flag_get(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	check DMA channel illegal flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check DMA0 channel0 illegal access flag is set or not */
FlagStatus state = RESET;
state = dma_illegal_access_flag_get (DMA0, DMA_CH0);
```

dma_illegal_access_flag_clear

The description of dma_illegal_access_flag_clear is shown as below:

Table 3-171. Function dma_illegal_access_flag_clear

Function name	dma_illegal_access_flag_clear
Function prototype	void dma_illegal_access_flag_clear(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	clear DMA channel illegal flag
Precondition	-
The called functions	-
Input parameter{in}	

dma_periph	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0...7)</i>	DMA channel selection, refer to Table 3-130. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMA0 channel0 illegal access flag */
dma_illegal_access_flag_clear (DMA0, DMA_CH0);
```

3.8. EFUSE

The Efuse controller has efuse macro that store system paramters. As a non-volatile unit of storage, the bit of efuse macro cannot be restored to 0 once it is programmed to 1. The EFUSE registers are listed in chapter [3.8.1](#), the EFUSE firmware functions are introduced in chapter [3.8.2](#).

3.8.1. Descriptions of Peripheral registers

EFUSE registers are listed in the table shown as below:

Table 3-172. EFUSE Registers

Registers	Descriptions
EFUSE_CS	control and status register
EFUSE_ADDR	address register
EFUSE_CTL	control register
EFUSE_TZCTL	trustzone control register
EFUSE_FP_CTL	flash protection control register
EFUSE_USER_CTL	user control register
EFUSE_MCU_INIT _DATAx(x = 0...2)	mcu initialization data register x(x = 0...2)
EFUSE_AES_KEYx (x = 0...3)	firmware AES key register x(x = 0...3)
EFUSE_ROTTPK_K EYx(x = 0...7)	RoTPK key register x(x = 0...7)
EFUSE_DPx(x = 0,1)	debug password register x(x = 0,1)
EFUSE_IAK_GSSA	IAK key or GSSA register x(x = 0...15)

Registers	Descriptions
x(x = 0...15)	
EFUSE_PUIDx(x = 0...3)	product UID register x(x = 0...3)
EFUSE_HUK_KEYx (x = 0...3)	HUK key register x(x = 0...3)
EFUSE_RF_DATAx (x = 0...11)	RF data register x(x = 0...11)
EFUSE_USER_DATAx(x = 0...7)	user data register x(x = 0...7)
EFUSE_PRE_TZEN	EFUSE Pre-TrustZone enable register
EFUSE_TZ_BOOT_ADDR	TrustZone boot address register
EFUSE_NTZ_BOOT_ADDR	No-TrustZone boot address register

3.8.2. Descriptions of Peripheral functions

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

Table 3-173. EFUSE firmware function

Function name	Function description
efuse_read	read EFUSE value
efuse_write	write EFUSE
efuse_boot_config	configure boot field
efuse_tz_control_config	configure trustzone control field
efuse_fp_config	configure flash protection field
efuse_mcu_init_data_write	write mcu initialization parameters
efuse_aes_key_write	write AES key
efuse_rotpk_key_write	write ROTPK key
efuse_dp_write	write debug password
efuse_iak_write	write IAK key
efuse_user_data_write	write user data
efuse_software_trustzone_enable	enable trustzone by software
efuse_software_trustzone_disable	disable trustzone by software
efuse_boot_address_get	get boot address information
efuse_flag_get	check EFUSE flag is set or not
efuse_flag_clear	clear EFUSE pending flag
efuse_interrupt_enable	enable EFUSE interrupt
efuse_interrupt_disable	disable EFUSE interrupt
efuse_interrupt_flag_get	check EFUSE interrupt flag is set or not
efuse_interrupt_flag_clear	clear EFUSE pending interrupt flag

Enum efuse_flag_enum

Table 3-174. Enum efuse_flag_enum

Member name	Function description
EFUSE_PGIF	programming operation completion flag
EFUSE_RDIF	read operation completion flag
EFUSE_OBERIF	overstep boundary error flag

Enum efuse_clear_flag_enum

Table 3-175. Enum efuse_clear_flag_enum

Member name	Function description
EFUSE_PGIC	clear programming operation completion flag
EFUSE_RDIC	clear read operation completion flag
EFUSE_OBERIC	clear overstep boundary error flag

Enum efuse_int_enum

Table 3-176. Enum efuse_int_enum

Member name	Function description
EFUSE_INTEN_PG	programming operation completion interrupt
EFUSE_INTEN_RD	read operation completion interrupt
EFUSE_INTEN_OBER	overstep boundary error interrupt

Enum efuse_int_flag_enum

Table 3-177. Enum efuse_int_flag_enum

Member name	Function description
EFUSE_INT_PGIF	programming operation completion interrupt flag
EFUSE_INT_RDIF	read operation completion interrupt flag
EFUSE_INT_OBERIF	overstep boundary error interrupt flag

Enum efuse_clear_int_flag_enum

Table 3-178. efuse_clear_int_flag_enum

Member name	Function description
EFUSE_INT_PGIC	clear programming operation completion interrupt flag
EFUSE_INT_RDIC	clear read operation completion interrupt flag
EFUSE_INT_OBERIC	clear overstep boundary error interrupt flag

Enum efuse_tz_enum

Table 3-179. Enum efuse_tz_enum

Member name	Function description
EFUSE_TZ	trustzone enable

EFUSE_N_TZ

trustzone disable

efuse_read

The description of efuse_read is shown as below:

Table 3-180. Function efuse_read

Function name	efuse_read
Function prototype	ErrStatus efuse_read(uint32_t ef_addr, uint32_t size, uint32_t buf[]);
Function descriptions	read EFUSE value
Precondition	-
The called functions	-
Input parameter{in}	
ef_addr	EFUSE address (0x00 – 0xf8)
Input parameter{in}	
size	size of EFUSE (0x01 – 0x40 bytes)
Output parameter{out}	
buf	the buffer for storing read-out EFUSE register value
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* read EFUSE USER DATA value */

uint32_t buffer[8] = {0};

ErrStatus flag = ERROR;

flag = efuse_read(0xd8, 32, buffer);
```

efuse_write

The description of efuse_write is shown as below:

Table 3-181. Function efuse_write

Function name	efuse_write
Function prototype	ErrStatus efuse_write(uint32_t ef_addr, uint32_t size, uint32_t buf[]);
Function descriptions	write EFUSE
Precondition	-
The called functions	-
Input parameter{in}	
ef_addr	EFUSE address (0x00 – 0xf8), when ef_addr is greater than 4, the ef_addr must be an integral multiple of 4
Input parameter{in}	
size	size of EFUSE (0x01 – 0x40 bytes)
Output parameter{out}	

buf	the buffer of data written to EFUSE
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write EFUSE USER DATA*/

uint32_t buffer[2] = {0x11223344, 0x55667788};

ErrStatus flag = ERROR;

flag = efuse_write(0xd8, 8, buffer);
```

efuse_boot_config

The description of efuse_boot_config is shown as below:

Table 3-182. Function efuse_boot_config

Function name	efuse_boot_config
Function prototype	ErrStatus efuse_boot_config(uint32_t size, uint8_t bt_value[]);
Function descriptions	boot configuration
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 1
Input parameter{in}	
bt_value	the value of boot configuration
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write Efuse control value */

uint8_t bt_value[1] = 0x32;

ErrStatus flag = ERROR;

flag = efuse_boot_config(1, bt_value);
```

efuse_tz_control_config

The description of efuse_tz_control_config is shown as below:

Table 3-183. Function efuse_tz_control_config

Function name	efuse_tz_control_config
Function prototype	ErrStatus efuse_tz_control_config(uint32_t size, uint8_t tz_ctl[]);

Function descriptions	trustzone control configuration
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 1
Input parameter{in}	
tz_ctl	trustzone control value
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write Trustzone control value */
uint8_t tz_ctl[1] = 0x32;

ErrStatus flag = ERROR;

flag = efuse_tz_control_config(1, tz_ctl);
```

efuse_fp_config

The description of efuse_fp_config is shown as below:

Table 3-184. Function efuse_fp_config

Function name	efuse_fp_config
Function prototype	ErrStatus efuse_fp_config(uint32_t size, uint8_t fp_value[]);
Function descriptions	flash protection configuration
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 1
Input parameter{in}	
fp_value	flash protection value
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write Flash protection value */
uint8_t fp_value[1] = 0x02;

ErrStatus flag = ERROR;
```

```
flag = efuse_fp_config(1, fp_value);
```

efuse_mcu_init_data_write

The description of efuse_mcu_init_data_write is shown as below:

Table 3-185. Function efuse_mcu_init_data_write

Function name	efuse_mcu_init_data_write
Function prototype	ErrStatus efuse_mcu_init_data_write(uint32_t size, uint8_t buf[]);
Function descriptions	write mcu initialization parameters
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 12
Input parameter{in}	
buf	the buffer of data written to efuse
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write MCU initialization parameters value */
```

```
uint32_t buffer[3] = {0x11223344, 0x55667788, 0x9900aabb};
```

```
ErrStatus flag = ERROR;
```

```
flag = efuse_mcu_init_data_write(12, (uint8_t *)buffer);
```

efuse_aes_key_write

The description of efuse_aes_key_write is shown as below:

Table 3-186. Function efuse_aes_key_write

Function name	efuse_aes_key_write
Function prototype	ErrStatus efuse_aes_key_write(uint32_t size, uint8_t buf[]);
Function descriptions	write AES key
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 16
Input parameter{in}	
buf	the buffer of data written to efuse
Output parameter{out}	
-	-

Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write AES key value */
uint32_t buffer[4] = {0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff};

ErrStatus flag = ERROR;

flag = efuse_aes_key_write(4, (uint8_t *)buffer);
```

efuse_rotpk_key_write

The description of efuse_rotpk_key_write is shown as below:

Table 3-187. Function efuse_rotpk_key_write

Function name	efuse_rotpk_key_write
Function prototype	ErrStatus efuse_rotpk_key_write(uint32_t size, uint8_t buf[]);
Function descriptions	write ROTPK key
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 32
Input parameter{in}	
buf	the buffer of data written to efuse
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write AES key value */
uint32_t buffer[8] = {0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff,
                      0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff};

ErrStatus flag = ERROR;

flag = efuse_rotpk_key_write(32, (uint8_t *)buffer);
```

efuse_dp_write

The description of efuse_dp_write is shown as below:

Table 3-188. Function efuse_dp_write

Function name	efuse_dp_write
---------------	----------------

Function prototype	ErrStatus efuse_dp_write(uint32_t size, uint8_t buf[]);
Function descriptions	write debug password
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 8
Input parameter{in}	
buf	the buffer of data written to efuse
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write Debug password value */
uint32_t buffer[2] = {0x11223344, 0x55667788};

ErrStatus flag = ERROR;

flag = efuse_dp_write(8, (uint8_t *)buffer);
```

efuse_iak_write

The description of efuse_iak_write is shown as below:

Table 3-189. Function efuse_iak_write

Function name	efuse_iak_write
Function prototype	ErrStatus efuse_iak_write(uint32_t size, uint8_t buf[]);
Function descriptions	write IAK key
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 64
Input parameter{in}	
buf	the buffer of data written to efuse
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write IAK value */
uint32_t buffer[16] = {0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff,
```



```
0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff,
0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff,
0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff };
```

```
ErrStatus flag = ERROR;
```

```
flag = efuse_iak_write(64, (uint8_t *)buffer);
```

efuse_user_data_write

The description of efuse_user_data_write is shown as below:

Table 3-190. Function efuse_user_data_write

Function name	efuse_user_data_write
Function prototype	ErrStatus efuse_user_data_write(uint32_t size, uint8_t buf[]);
Function descriptions	write user data
Precondition	-
The called functions	efuse_write
Input parameter{in}	
size	size of data (byte), the size must be 32
Input parameter{in}	
buf	the buffer of data written to efuse
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* write User data value */
```

```
uint32_t buffer[8] = {0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff
0x11223344, 0x55667788, 0x9900aabb, 0xccddeeff};
```

```
ErrStatus flag = ERROR;
```

```
flag = efuse_user_data_write(32, (uint8_t *)buffer);
```

efuse_software_trustzone_enable

The description of efuse_software_trustzone_enable is shown as below:

Table 3-191. Function efuse_software_trustzone_enable

Function name	efuse_software_trustzone_enable
Function prototype	void efuse_software_trustzone_enable(void);
Function descriptions	enable trustzone by software
Precondition	-

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

Example:

```
/* enable Trustzone function by software */
```

```
efuse_software_trustzone_enable();
```

efuse_software_trustzone_disable

The description of efuse_software_trustzone_disable is shown as below:

Table 3-192. Function efuse_software_trustzone_disable

Function name	efuse_software_trustzone_disable
Function prototype	void efuse_software_trustzone_disable(void);
Function descriptions	disable trustzone by software
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable Trustzone function by software */
```

```
efuse_software_trustzone_disable();
```

efuse_boot_address_get

The description of efuse_boot_address_get is shown as below:

Table 3-193. Function efuse_boot_address_get

Function name	efuse_boot_address_get
Function prototype	uint32_t efuse_boot_address_get(efuse_tz_enum tz);
Function descriptions	get boot address information
Precondition	-
The called functions	-
Input parameter{in}	

tz	specifies the current trustzone state, refer to Table 3-179. Enum efuse_tz_enum
<i>EFUSE_TZ</i>	trustzone enable
<i>EFUSE_N_TZ</i>	trustzone disable
Output parameter{out}	
-	-
Return value	
uint32_t	current boot address (0x0 – 0xFFFFFFFF)

Example:

```
/* get boot address when Trustzone disable */
uint32_t addr = 0;
addr = effuse_boot_address_get(EFUSE_N_TZ);
```

efuse_flag_get

The description of efuse_flag_get is shown as below:

Table 3-194. Function efuse_flag_get

Function name	efuse_flag_get
Function prototype	FlagStatus efuse_flag_get(efuse_flag_enum efuse_flag);
Function descriptions	check EFUSE flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
efuse_flag	specifies to get a flag, refer to Table 3-174. Enum efuse_flag_enum
<i>EFUSE_PGIF</i>	programming operation completion flag
<i>EFUSE_RDIF</i>	read operation completion flag
<i>EFUSE_OBERIF</i>	overstep boundary error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get EFUSE write operation complete flag status */
FlagStatus state = efuse_flag_get(EFUSE_PGIF);
```

efuse_flag_clear

The description of efuse_flag_clear is shown as below:

Table 3-195. Function efuse_flag_clear

Function name	efuse_flag_clear
Function prototype	void efuse_flag_clear(efuse_clear_flag_enum efuse_cflag);
Function descriptions	clear EFUSE pending flag
Precondition	-
The called functions	-
Input parameter{in}	
efuse_cflag	specifies to clear a flag, refer to Table 3-175. Enum efuse_clear_flag_enum
<i>EFUSE_PGIC</i>	clear programming operation completion flag
<i>EFUSE_RDIC</i>	clear read operation completion flag
<i>EFUSE_OBERIC</i>	clear overstep boundary error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear EFUSE write operation complete flag status */
efuse_flag_clear(EFUSE_PGIC);
```

efuse_interrupt_enable

The description of efuse_interrupt_enable is shown as below:

Table 3-196. Function efuse_interrupt_enable

Function name	efuse_interrupt_enable
Function prototype	void efuse_interrupt_enable(efuse_int_enum source);
Function descriptions	enable EFUSE interrupt
Precondition	-
The called functions	-
Input parameter{in}	
source	specifies an interrupt to enable, refer to Table 3-176. Enum efuse_int_enum
<i>EFUSE_INTEN_PG</i>	programming operation completion interrupt
<i>EFUSE_INTEN_RD</i>	read operation completion interrupt
<i>EFUSE_INTEN_OBER</i>	overstep boundary error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable EFUSE write operation complete interrupt */
```

```
efuse_interrupt_enable(EFUSE_INTEN_PG);
```

efuse_interrupt_disable

The description of efuse_interrupt_disable is shown as below:

Table 3-197. Function efuse_interrupt_disable

Function name	efuse_interrupt_disable
Function prototype	void efuse_interrupt_disable(efuse_int_enum source);
Function descriptions	disable EFUSE interrupt
Precondition	-
The called functions	-
Input parameter{in}	
source	specifies an interrupt to disable, refer to Table 3-176. Enum efuse_int_enum
EFUSE_INTEN_PG	programming operation completion interrupt
EFUSE_INTEN_RD	read operation completion interrupt
EFUSE_INTEN_OBER	overstep boundary error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable EFUSE write operation complete interrupt */
```

```
efuse_interrupt_disable(EFUSE_INTEN_PG);
```

efuse_interrupt_flag_get

The description of efuse_interrupt_flag_get is shown as below:

Table 3-198. Function efuse_interrupt_flag_get

Function name	efuse_interrupt_flag_get
Function prototype	FlagStatus efuse_interrupt_flag_get(efuse_int_flag_enum int_flag);
Function descriptions	check EFUSE interrupt flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	specifies to get a flag, refer to Table 3-177. Enum efuse_int_flag_enum
EFUSE_INT_PGIF	programming operation completion interrupt flag
EFUSE_INT_RDIF	read operation completion interrupt flag
EFUSE_INT_OBERIF	overstep boundary error interrupt flag
Output parameter{out}	

-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get EFUSE write operation complete interrupt flag status */
```

```
FlagStatus state = efuse_interrupt_flag_get(EFUSE_INT_PGIF);
```

efuse_interrupt_flag_clear

The description of efuse_interrupt_flag_clear is shown as below:

Table 3-199. Function efuse_interrupt_flag_clear

Function name	efuse_interrupt_flag_clear
Function prototype	void efuse_interrupt_flag_clear(efuse_clear_int_flag_enum int_cflag);
Function descriptions	clear EFUSE pending interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_cflag	specifies to clear a flag, refer to Table 3-177. Enum efuse_int_flag_enum
<i>EFUSE_INT_PGIC</i>	clear programming operation completion interrupt flag
<i>EFUSE_INT_RDIC</i>	clear read operation completion interrupt flag
<i>EFUSE_INT_OBERIC</i>	clear overstep boundary error interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear EFUSE write operation complete interrupt flag status */
```

```
efuse_interrupt_flag_clear(EFUSE_INT_PGIC);
```

3.9. EXTI

EXTI is the interrupt / event controller in the MCU. It contains up to 29 independent edge detectors and generates interrupt requests or events to the processor. The EXTI registers are listed in chapter [3.9.1](#), the EXTI firmware functions are introduced in chapter [3.9.2](#).

3.9.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

Table 3-200. 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
EXTI_SECCFG	Security configuration register
EXTI_PRIVCFG	Privilege configuration register
EXTI_LOCK	Lock register

3.9.2. Descriptions of Peripheral functions

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

Table 3-201. EXTI firmware function

Function name	Function description
exti_deinit	deinitialize the EXTI, reset the value of all EXTI registers into initial values
exti_init	initialize the EXTI, enable the configuration of EXTI initialize
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_security_enable	enable the security attribution from EXTI line x
exti_security_disable	disable the security attribution from EXTI line x
exti_privilege_enable	enable the privileged access from EXTI line x
exti_privilege_disable	disable the privileged access from EXTI line x
exti_lock_enable	lock EXTI security attribution and privileged access configuration
exti_software_interrupt_enable	enable the EXTI software interrupt event
exti_software_interrupt_disable	disable the EXTI software interrupt event
exti_flag_get	get EXTI line x pending flag
exti_flag_clear	clear EXTI line x pending flag
exti_interrupt_flag_get	get EXTI line x flag when the interrupt flag is set
exti_interrupt_flag_clear	clear EXTI line x pending flag

Enum exti_line_enum

Table 3-202. Enum exti_line_enum

Member name	Function description
EXTI_0	EXTI line 0
EXTI_1	EXTI line 1

Member name	Function description
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
EXTI_19	EXTI line 19
EXTI_20	EXTI line 20
EXTI_21	EXTI line 21
EXTI_22	EXTI line 22
EXTI_23	EXTI line 23
EXTI_24	EXTI line 24
EXTI_25	EXTI line 25
EXTI_26	EXTI line 26
EXTI_27	EXTI line 27
EXTI_28	EXTI line 28

Enum exti_mode_enum

Table 3-203. Enum exti_mode_enum

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

Enum exti_trig_type_enum

Table 3-204. 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 3-205. Function exti_deinit

Function name	exti_deinit
Function prototype	void exti_deinit(void);
Function descriptions	deinitialize the EXTI, reset the value of all EXTI registers into initial values
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 3-206. 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, enable the configuration of EXTI initialize
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Input parameter{in}	
mode	EXTI mode, refer to Table 3-203. Enum exti_mode_enum
<i>EXTI_INTERRUPT</i>	interrupt mode
<i>EXTI_EVENT</i>	event mode
Input parameter{in}	
trig_type	trigger type, refer to Table 3-204. Enum exti_trig_type_enum
<i>EXTI_TRIG_RISING</i>	rising edge trigger

<i>EXTI_TRIG_FALLING</i>	falling edge trigger
<i>EXTI_TRIG_BOTH</i>	rising edge and falling edge trigger
<i>EXTI_TRIG_NONE</i>	without rising edge or falling edge trigger
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 3-207. 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 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
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 3-208. 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 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
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 3-209. 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 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
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 3-210. 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 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the events from EXTI line 0 */
```

```
exti_event_disable(EXTI_0);
```

exti_security_enable

The description of exti_security_enable is shown as below:

Table 3-211. Function exti_security_enable

Function name	exti_security_enable
Function prototype	void exti_security_enable(exti_line_enum linex);
Function descriptions	enable the security attribution from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the security attribution */
```

```
exti_security_enable(EXTI_0);
```

exti_security_disable

The description of exti_security_disable is shown as below:

Table 3-212. Function exti_security_disable

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

Input parameter{in}	
linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the security attribution */
```

```
exti_security_disable(EXTI_0);
```

exti_privilege_enable

The description of exti_privilege_enable is shown as below:

Table 3-213. Function exti_privilege_enable

Function name	exti_privilege_enable
Function prototype	void exti_privilege_enable(exti_line_enum linex);
Function descriptions	enable the privileged access from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the privileged access */
```

```
exti_privilege_enable(EXTI_0);
```

exti_privilege_disable

The description of exti_privilege_disable is shown as below:

Table 3-214. Function exti_privilege_disable

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

Input parameter{in}	
linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the privileged access */
exti_privilege_disable(EXTI_0);
```

exti_lock_enable

The description of exti_lock_enable is shown as below:

Table 3-215. Function exti_lock_enable

Function name	exti_lock_enable
Function prototype	void exti_lock_enable(void);
Function descriptions	lock EXTI security attribution and privileged access configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable lock EXTI security attribution and privileged access configuration */
exti_lock_enable();
```

exti_software_interrupt_enable

The description of exti_software_interrupt_enable is shown as below:

Table 3-216. 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 EXTI software interrupt event
Precondition	-
The called functions	-
Input parameter{in}	

linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable EXTI line 0 software interrupt event */
```

```
exti_software_interrupt_enable(EXTI_0);
```

exti_software_interrupt_disable

The description of exti_software_interrupt_disable is shown as below:

Table 3-217. 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 EXTI software interrupt event
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable EXTI line 0 software interrupt event */
```

```
exti_software_interrupt_disable(EXTI_0);
```

exti_flag_get

The description of exti_flag_get is shown as below:

Table 3-218. 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 pending flag
Precondition	-
The called functions	-
Input parameter{in}	

linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
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 3-219. 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 pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
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 3-220. 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 flag when the interrupt flag is set
Precondition	-
The called functions	-
Input parameter{in}	

linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
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 3-221. 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 pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-202. Enum exti_line_enum
<i>EXTI_x</i>	x=0,1,2..28
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear EXTI line 0 interrupt flag status */
```

```
exti_interrupt_flag_clear(EXTI_0);
```

3.10. FMC

There is flash controller and option byte for GD32W51x series. The FMC registers are listed in chapter [3.10.1](#) the FMC firmware functions are introduced in chapter [3.10.2](#).

3.10.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

Table 3-222. FMC Registers

Registers	Descriptions
FMC_KEY	unlock key register
FMC_OBKEY	option bytes unlock key register
FMC_STAT	status register
FMC_CTL	control register
FMC_ADDR	address register
FMC_OBSTAT	option byte status register
FMC_SECKEY	secure unlock key register
FMC_SECSTAT	secure status register
FMC_SECCTL	secure control register
FMC_SECADDR	secure address register
FMC_OBR	option byte register
FMC_OBUSER	option byte user register
FMC_SECMCFG0	secure mark configuration register 0
FMC_DMP0	secure dedicated mark protection register 0
FMC_OBWRP0	option byte write protection area register 0
FMC_SECMCFG1	secure mark configuration register 1
FMC_DMP1	secure dedicated mark protection register 1
FMC_OBWRP1	option byte write protection area register 1
FMC_SECMCFG2	secure mark configuration register 2
FMC_SECMCFG3	secure mark configuration register 3
FMC_NODEC0	NO-RTDEC region register 0
FMC_NODEC1	NO-RTDEC region register 1
FMC_NODEC2	NO-RTDEC region register 2
FMC_NODEC3	NO-RTDEC region register 3
FMC_OFRG	offset region register
FMC_OFVR	offset value register
FMC_DMPCTL	DMP control register
FMC_PRIVCFG	privilege configuration register
FMC_PID	Product ID register

3.10.2. Descriptions of Peripheral functions

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

Table 3-223. FMC firmware function

Function name	Function description
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_word_program	FMC program a word at the corresponding address

Function name	Function description
fmc_continuous_program	FMC program continuously at the corresponding address
sram1_reset_enable	enable SRAM1 reset automatically function
sram1_reset_disable	disable SRAM1 reset automatically function
fmc_privilege_enable	enable the privileged access
fmc_privilege_disable	disable the privileged access
ob_unlock	unlock the option byte operation
ob_lock	lock the option byte operation
ob_start	send option bytes modification start command
ob_reload	reload option bytes
ob_security_protection_config	configure the option bytes security protection
ob_trustzone_enable	enable trustzone
ob_trustzone_disable	disable trustzone
ob_user_write	program option bytes USER
ob_write_protection_config	configure write protection pages
ob_secmark_config	configure secure mark pages
ob_dmp_access_enable	enable DMP region access right
ob_dmp_access_disable	disable DMP region access right
ob_dmp_config	configure DMP secure pages
ob_dmp_enable	enable DMP function
fmc_no_rtdec_config	configure NO-RTDEC pages
fmc_offset_region_config	configure offset region
fmc_offset_value_config	configure offset value
ob_write_protection_get	get the value of option bytes write protection state, only applies to get the status of write/erase protection setting by Efuse
ob_user_get	get the value of option bytes USER
ob_security_protection_flag_get	get option bytes security protection state
ob_trustzone_state_get	get option bytes trustzone state
ob_memory_mode_state_get	get the state of MCU memory structure is FMC mode or QSPI mode
ob_exist_state_get	get the state of whether the option byte exist or not
fmc_flag_get	get FMC flag status
fmc_flag_clear	clear the FMC flag
fmc_interrupt_enable	enable FMC interrupt
fmc_interrupt_disable	disable FMC interrupt
fmc_interrupt_flag_get	get FMC interrupt flag
fmc_interrupt_flag_clear	clear FMC interrupt flag

fmc_state_enum

Table 3-224. fmc_state_enum

enum name	enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error
FMC_OBERR	option bytes error
FMC_SECERR	secure error, only available in secure mode

fmc_unlock

The description of fmc_unlock is shown as below:

Table 3-225. 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 3-226. 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 3-227. Function fmc_page_erase

Function name	fmc_page_erase
Function prototype	fmc_state_enum fmc_page_erase(uint32_t page_address);
Function descriptions	FMC erase page
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
page_address	target page address
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* erase page */
```

```
fmc_state_enum state;
```

```
fmc_unlock();
```

```
state = fmc_page_erase ( 0x08004000);
```

fmc_mass_erase

The description of fmc_mass_erase is shown as below:

Table 3-228. 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	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* erase the whole chip */

fmc_unlock();

fmc_state_enum state;

state = fmc_mass_erase();
```

fmc_word_program

The description of fmc_word_program is shown as below:

Table 3-229. 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, fmc_page_erase/fmc_mass_erase
The called functions	-
Input parameter{in}	
address	the address to program
data	word to program(0x00000000 - 0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

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

fmc_state_enum state;

fmc_unlock();

fmc_page_erase(0x08004000);

state = fmc_word_program (0x08004000, 0xaabbccdd);
```

fmc_continuous_program

The description of fmc_continuous_program is shown as below:

Table 3-230. Function fmc_continuous_program

Function name	fmc_continuous_program
Function prototype	fmc_state_enum fmc_continuous_program(uint32_t address, uint32_t data[], uint32_t size);
Function descriptions	FMC program data continuously at the corresponding address
Precondition	fmc_unlock, fmc_page_erase/fmc_mass_erase
The called functions	-
Input parameter{in}	
address	address to program, must be 4-byte aligned
Input parameter{in}	
data[]	data buffer to program
Input parameter{in}	
size	data buffer size in bytes, must be 4-byte aligned
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* program data continuously at the corresponding address */
```

```
uint32_t data[16]= {0x01234567, 0x01234567, 0x01234567, 0x01234567, 0x01234567,
0x01234567, 0x01234567, 0x01234567, 0x01234567, 0x01234567};
```

```
fmc_state_enum state;
```

```
fmc_unlock();
```

```
state = fmc_word_program (0x08004000, data[],16);
```

sram1_reset_enable

The description of sram1_reset_enable is shown as below:

Table 3-231. Function sram1_reset_enable

Function name	sram1_reset_enable
Function prototype	void sram1_reset_enable(void);
Function descriptions	enable SRAM1 reset automatically function
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SRAM1 reset automatically function */

ob_unlock();

sram1_reset_enable();
```

sram1_reset_disable

The description of sram1_reset_disable is shown as below:

Table 3-232. Function sram1_reset_disable

Function name	sram1_reset_disable
Function prototype	void sram1_reset_disable(void);
Function descriptions	disable SRAM1 reset automatically function
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SRAM1 reset automatically function */

ob_unlock();

sram1_reset_disable();
```

fmc_privilege_enable

The description of fmc_privilege_enable is shown as below:

Table 3-233. Function fmc_privilege_enable

Function name	fmc_privilege_enable
Function prototype	void fmc_privilege_enable(void);
Function descriptions	enable the privileged access
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable the privileged access */
```

```
fmc_privilege_enable();
```

fmc_privilege_disable

The description of fmc_privilege_disable is shown as below:

Table 3-234. Function fmc_privilege_disable

Function name	fmc_privilege_disable
Function prototype	void fmc_privilege_disable(void);
Function descriptions	disable the privileged access
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the privileged access */
```

```
fmc_privilege_disable();
```

ob_unlock

The description of ob_unlock is shown as below:

Table 3-235. Function ob_unlock

Function name	ob_unlock
Function prototype	void ob_unlock(void);
Function descriptions	unlock the option byte operation
Precondition	-
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 3-236. Function ob_lock

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

Example:

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

```
ob_lock();
```

ob_start

The description of ob_start is shown as below:

Table 3-237. Function ob_start

Function name	ob_start
Function prototype	void ob_start(void);
Function descriptions	send option bytes modification start command
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* program option bytes USER data */
```

```
ob_unlock( );
```

```
ob_user_write(0xFFFF);
```

```
ob_start();
```

ob_reload

The description of ob_reload is shown as below:

Table 3-238. Function ob_reload

Function name	ob_reload
Function prototype	void ob_reload(void);
Function descriptions	reload option bytes
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* program option bytes USER data */
```

```
ob_unlock();
```

```
ob_user_write(0xFFFF);
```

```
ob_start();
```

```
ob_reload();
```

ob_security_protection_config

The description of ob_security_protection_config is shown as below:

Table 3-239. 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	-
Input parameter{in}	
ob_spc	specify security protection

<i>FMC_NSPC</i>	no security protection
<i>FMC_SPC_P0_5</i>	security protection level 0.5
<i>FMC_SPC_P1</i>	security protection level 1
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* configure low security protection */

ob_unlock();

ob_security_protection_config (FMC_NSPC);

ob_start();
```

ob_trustzone_enable

The description of ob_trustzone_enable is shown as below:

Table 3-240. Function ob_trustzone_enable

Function name	ob_trustzone_enable
Function prototype	fmc_state_enum ob_trustzone_enable(void);
Function descriptions	enable trustzone
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* enable trustzone */

ob_unlock();

ob_trustzone_enable();
```

ob_trustzone_disable

The description of ob_trustzone_disable is shown as below:

Table 3-241. Function ob_trustzone_disable

Function name	ob_trustzone_disable
----------------------	----------------------

Function prototype	ErrStatus ob_trustzone_disable(void);
Function descriptions	disable trustzone
Precondition	ob_unlock
The called functions	ob_security_protection_flag_get
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* disable trustzone */

ErrStatus state;

ob_unlock();

state = ob_trustzone_disable();
```

ob_user_write

The description of ob_user_write is shown as below:

Table 3-242. Function ob_user_write

Function name	ob_user_write
Function prototype	fmc_state_enum ob_user_write(uint16_t ob_user);
Function descriptions	program option bytes USER
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_user	option bytes user value
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* program option bytes USER data */

ob_unlock();

ob_user_write(0xFFFF);

ob_start();
```

ob_write_protection_config

The description of ob_write_protection_config is shown as below:

Table 3-243. Function ob_write_protection_config

Function name	ob_write_protection_config
Function prototype	fmc_state_enum ob_write_protection_config(uint32_t wrp_spage, uint32_t wrp_epage, uint32_t wrp_register_index);
Function descriptions	configure write protection pages
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
wrp_spage	start page of write protection area, 0~512
Input parameter{in}	
wrp_epage	end page of write protection area, 0~512
Input parameter{in}	
wrp_register_index	FMC_OBWRP _x register index
<i>OBWRP_INDEX0</i>	option byte write protection area register 0
<i>OBWRP_INDEX1</i>	option byte write protection area register 1
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* configure write protection pages */
```

```
ob_unlock();
```

```
ob_write_protection_config(WRP_REGION_SPAGE, WRP_REGION_EPAGE, OBWRP_INDEX0);
```

```
ob_start();
```

ob_secmark_config

The description of ob_secmark_config is shown as below:

Table 3-244. Function ob_secmark_config

Function name	ob_secmark_config
Function prototype	void ob_secmark_config(uint32_t secm_spage, uint32_t secm_epage, uint32_t secm_register_index);
Function descriptions	configure secure mark pages
Precondition	ob_unlock
The called functions	-

Input parameter{in}	
secm_spage	start page of secure mark area, 0~0x03FF
Input parameter{in}	
secm_epage	end page of secure mark area, 0~0x03FF
Input parameter{in}	
secm_register_index	secure mark register index
<i>SECM_INDEX0</i>	secure mark configuration register 0
<i>SECM_INDEX1</i>	secure mark configuration register 1
<i>SECM_INDEX2</i>	secure mark configuration register 2
<i>SECM_INDEX3</i>	secure mark configuration register 3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure secure mark pages */
```

```
ob_unlock();
```

```
ob_secmark_config(SECM_REGION_SPAGE,SECM_REGION_EPAGE,SECM_INDEX3);
```

ob_dmp_access_enable

The description of ob_dmp_access_enable is shown as below:

Table 3-245. Function ob_dmp_access_enable

Function name	ob_dmp_access_enable
Function prototype	void ob_dmp_access_enable(uint32_t DMP_register_index);
Function descriptions	enable DMP region access right
Precondition	-
The called functions	-
Input parameter{in}	
dmp_register_index	DMP region configuration register index
<i>DMP_INDEX0</i>	secure DMP configuration register 0
<i>DMP_INDEX1</i>	secure DMP configuration register 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMP region access right */
```

```
ob_dmp_access_enable(DMP_INDEX0);
```

ob_dmp_access_disable

The description of ob_dmp_access_disable is shown as below:

Table 3-246. Function ob_dmp_access_disable

Function name	ob_dmp_access_disable
Function prototype	void ob_dmp_access_disable(uint32_t DMP_register_index);
Function descriptions	disable DMP region access right
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
dmp_register_index	DMP region configuration register index
<i>DMP_INDEX0</i>	secure DMP configuration register 0
<i>DMP_INDEX1</i>	secure DMP configuration register 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMP region access right */
ob_unlock();
ob_dmp_access_disable(DMP_INDEX0);
```

ob_dmp_config

The description of ob_dmp_config is shown as below:

Table 3-247. Function ob_dmp_config

Function name	ob_dmp_config
Function prototype	ErrStatus ob_dmp_config(uint32_t dmp_epage, uint32_t dmp_register_index);
Function descriptions	configure DMP pages
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
dmp_epage	end page of secure DMP mark area 0~0x03FF
Input parameter{in}	
dmp_register_index	DMP region register index
<i>DMP_INDEX0</i>	secure DMP configuration register 0
<i>DMP_INDEX1</i>	secure DMP configuration register 1
Output parameter{out}	
-	-

Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* configure DMP pages */

ob_unlock();

ob_dmp_config(DMP_REGION_EPAGE, DMP_INDEX0);
```

ob_dmp_enable

The description of ob_dmp_enable is shown as below:

Table 3-248. Function ob_dmp_enable

Function name	ob_dmp_enable
Function prototype	fmc_state_enum ob_dmp_enable(uint32_t dmp_register_index);
Function descriptions	enable DMP function
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
dmp_register_index	DMP region register index
DMP_INDEX0	secure DMP configuration register 0
DMP_INDEX1	secure DMP configuration register 1
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* enable DMP function */

ob_unlock();

ob_dmp_enable(DMP_INDEX0);
```

ob_dmp_disable

The description of ob_dmp_disable is shown as below:

Table 3-249. Function ob_dmp_disable

Function name	ob_dmp_disable
Function prototype	fmc_state_enum ob_dmp_disable(uint32_t dmp_register_index);
Function descriptions	disable DMP function
Precondition	ob_unlock
The called functions	-
Input parameter{in}	

dmp_register_index	DMP region register index
<i>DMP_INDEX0</i>	secure DMP configuration register 0
<i>DMP_INDEX1</i>	secure DMP configuration register 1
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to Table 3-224. fmc_state_enum

Example:

```
/* disable DMP function */

ob_unlock();

ob_dmp_disable(DMP_INDEX0);
```

fmc_no_rtdec_config

The description of fmc_no_rtdec_config is shown as below:

Table 3-250. Function fmc_no_rtdec_config

Function name	fmc_no_rtdec_config
Function prototype	void fmc_no_rtdec_config(uint32_t nodec_spage, uint32_t nodec_epage, uint32_t nodec_register_index);
Function descriptions	configure NO-RTDEC pages
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
nodec_spage	start page of NO-RTDEC area, 0~0x03FF
Input parameter{in}	
nodec_epage	end page of NO-RTDEC area, 0~0x03FF
Input parameter{in}	
nodec_register_index	NO-RTDEC region register index
<i>NODEC_INDEX0</i>	NO-RTDEC region register 0
<i>NODEC_INDEX1</i>	NO-RTDEC region register 1
<i>NODEC_INDEX2</i>	NO-RTDEC region register 2
<i>NODEC_INDEX3</i>	NO-RTDEC region register 3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure NO-RTDEC pages */

ob_unlock();
```

```
fmc_no_rtdec_config(NODEC_REGION_SPAGE, NODEC_REGION_EPAGE, NODEC_IN
DEX0);
```

fmc_offset_region_config

The description of fmc_offset_region_config is shown as below:

Table 3-251. Function fmc_offset_region_config

Function name	fmc_offset_region_config
Function prototype	void fmc_offset_region_config(uint32_t of_spage, uint32_t of_epage);
Function descriptions	configure offset region
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
of_spage	start page of offset region, 0~0x1FFF
Input parameter{in}	
of_epage	end page of offset region, 0~0x1FFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure offset region */
```

```
ob_unlock();
```

```
fmc_offset_region_config(SOURCE_START_PAGE, SOURCE_END_PAGE);
```

fmc_offset_value_config

The description of fmc_offset_value_config is shown as below:

Table 3-252. Function fmc_offset_value_config

Function name	fmc_offset_value_config
Function prototype	void fmc_offset_value_config(uint32_t of_value);
Function descriptions	configure offset value
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
of_value	offset value, 0~0x1FFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure offset value */

ob_unlock();

fmc_offset_value_config(PAGE_OFFSET_VALUE);
```

ob_write_protection_get

The description of ob_write_protection_get is shown as below:

Table 3-253. Function ob_write_protection_get

Function name	ob_write_protection_get
Function prototype	FlagStatus ob_write_protection_get(void);
Function descriptions	get write protection state, only applies to get the status of write/erase protection setting by EFUSE
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the FMC write protection status */

FlagStatus status;

status = ob_write_protection_get();
```

ob_user_get

The description of ob_user_get is shown as below:

Table 3-254. Function ob_user_get

Function name	ob_user_get
Function prototype	uint16_t ob_user_get(void);
Function descriptions	get the value of option bytes USER
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
uint16_t	0x0000~0xFFFF

Example:

```
/* get the value of option bytes USER */
```

```
uint16_t user_value;
```

```
user_value = ob_user_get();
```

ob_security_protection_flag_get

The description of ob_security_protection_flag_get is shown as below:

Table 3-255. Function ob_security_protection_flag_get

Function name	ob_security_protection_flag_get
Function prototype	FlagStatus ob_security_protection_flag_get(uint32_t spc_state);
Function descriptions	get the FMC option bytes security protection state
Precondition	-
The called functions	-
Input parameter{in}	
spc_state	security protection level
OB_FLAG_NSPC	option bytes security protection level 0
OB_FLAG_SPC0_5	option bytes security protection level 0.5
OB_FLAG_SPC1	option bytes security protection level 1
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check FMC option bytes security protection level 1 is set or not */
```

```
FlagStatus status;
```

```
status = ob_security_protection_flag_get(OB_FLAG_SPC1);
```

ob_trustzone_state_get

The description of ob_trustzone_state_get is shown as below:

Table 3-256. Function ob_trustzone_state_get

Function name	ob_trustzone_state_get
Function prototype	FlagStatus ob_trustzone_state_get(void);
Function descriptions	get trustzone state
Precondition	-
The called functions	-

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

Example:

```
/* get trustzone state */
```

```
FlagStatus status;
```

```
status = ob_trustzone_state_get();
```

ob_memory_mode_state_get

The description of ob_memory_mode_state_get is shown as below:

Table 3-257. Function ob_memory_mode_state_get

Function name	ob_memory_mode_state_get
Function prototype	FlagStatus ob_memory_mode_state_get(void);
Function descriptions	get the state of MCU memory structure is FMC mode or QSPI mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the state of MCU memory structure is FMC mode or QSPI mode */
```

```
FlagStatus status;
```

```
status = ob_memory_mode_state_get();
```

ob_exist_state_get

The description of ob_exist_state_get is shown as below:

Table 3-258. Function ob_exist_state_get

Function name	ob_exist_state_get
Function prototype	FlagStatus ob_exist_state_get(void);
Function descriptions	get the state of whether the option byte exist or not
Precondition	-

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

Example:

```
/* get the state of whether the option byte exist or not */
```

```
FlagStatus status;
```

```
status = ob_exist_state_get();
```

fmc_flag_get

The description of fmc_flag_get is shown as below:

Table 3-259. 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	FMC flag
<i>FMC_FLAG_BUSY</i>	FMC busy flag bit
<i>FMC_FLAG_OBERR</i>	FMC option bytes error flag bit
<i>FMC_FLAG_WPERR</i>	FMC erase/program protection error flag bit
<i>FMC_FLAG_END</i>	FMC end of operation flag bit
<i>FMC_FLAG_SECERR</i>	FMC secure error flag bit
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check the FMC_FLAG_END flag set or not*/
```

```
FlagStatus flag;
```

```
flag = fmc_flag_get(FMC_FLAG_END);
```

fmc_flag_clear

The description of fmc_flag_clear is shown as below:

Table 3-260. 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	FMC flag
<i>FMC_FLAG_OBERR</i>	FMC option bytes error flag bit
<i>FMC_FLAG_WPERR</i>	FMC erase/program protection error flag bit
<i>FMC_FLAG_END</i>	FMC end of operation flag bit
<i>FMC_FLAG_SECERR</i>	FMC secure error flag bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the FMC_FLAG_END flag */
fmc_flag_clear(FMC_FLAG_END);
```

fmc_interrupt_enable

The description of fmc_interrupt_enable is shown as below:

Table 3-261. Function fmc_interrupt_enable

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

Example:

```
/* enable FMC interrupt */

fmc_unlock();

fmc_interrupt_enable(FMC_INT_END);
```

fmc_interrupt_disable

The description of fmc_interrupt_disable is shown as below:

Table 3-262. Function fmc_interrupt_disable

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

Example:

```
/* disable FMC interrupt */

fmc_unlock();

fmc_interrupt_disable(FMC_INT_END);
```

fmc_interrupt_flag_get

The description of fmc_interrupt_flag_get is shown as below:

Table 3-263. Function fmc_interrupt_flag_get

Function name	fmc_interrupt_flag_get
Function prototype	FlagStatus fmc_interrupt_flag_get(uint32_t flag);
Function descriptions	get FMC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	the FMC interrupt flag
<i>FMC_INT_FLAG_WPE</i>	FMC secure/non-secure erase/program protection error interrupt flag

<i>RR</i>	
<i>FMC_INT_FLAG_END</i>	FMC secure/non-secure end of operation interrupt flag
<i>FMC_INT_FLAG_SEC ERR</i>	FMC secure error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check operation interrupt flag is set or not */
```

```
FlagStatus flag;
```

```
flag = fmc_interrupt_flag_get(FMC_INT_FLAG_END);
```

fmc_interrupt_flag_clear

The description of fmc_interrupt_flag_clear is shown as below:

Table 3-264. Function fmc_interrupt_flag_clear

Function name	fmc_interrupt_flag_clear
Function prototype	void fmc_interrupt_flag_clear(uint32_t flag);
Function descriptions	clear FMC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	the FMC interrupt flag
<i>FMC_INT_FLAG_WPE RR</i>	FMC secure/non-secure erase/program protection error interrupt flag
<i>FMC_INT_FLAG_END</i>	FMC secure/non-secure end of operation interrupt flag
<i>FMC_INT_FLAG_SEC ERR</i>	FMC secure error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear operation interrupt flag */
```

```
fmc_interrupt_flag_clear(FMC_INT_END);
```

3.11. 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 [3.11.1](#) the FWDGT firmware functions are introduced in chapter [3.11.2](#).

3.11.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

Table 3-265. FWDGT Registers

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

3.11.2. Descriptions of Peripheral functions

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

Table 3-266. 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 FWDGT counter
fwdgt_prescaler_value_config	configure the FWDGT counter prescaler value
fwdgt_reload_value_config	configure the FWDGT 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 3-267. Function fwdgt_write_enable

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 3-268. 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 3-269. Function fwdgt_enable

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

-	-
Return value	
-	-

Example:

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

```
wdgt_enable ( );
```

wdgt_prescaler_value_config

The description of wdgt_prescaler_value_config is shown as below:

Table 3-270. Function wdgt_prescaler_value_config

Function name	wdgt_prescaler_value_config
Function prototype	ErrStatus wdgt_prescaler_value_config(uint16_t prescaler_value);
Function descriptions	configure the FWDGT counter prescaler value
Precondition	-
The called functions	-
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 or SUCCESS

Example:

```
/* configure the FWDGT counter prescaler value */
```

```
wdgt_prescaler_value_config (FWDGT_PSC_DIV8);
```

wdgt_reload_value_config

The description of wdgt_reload_value_config is shown as below:

Table 3-271. Function wdgt_reload_value_config

Function name	wdgt_reload_value_config
Function prototype	ErrStatus wdgt_reload_value_config(uint16_t reload_value);
Function descriptions	configure the FWDGT counter reload value

Precondition	-
The called functions	-
Input parameter{in}	
reload_value	specify reload value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* configure the FWDGT counter reload value */
```

```
fwdgt_reload_value_config(625);
```

fwdgt_counter_reload

The description of fwdgt_counter_reload is shown as below:

Table 3-272. 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 3-273. 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
<i>FWDGT_PSC_DIV4</i>	FWDGT prescaler set to 4
<i>FWDGT_PSC_DIV8</i>	FWDGT prescaler set to 8
<i>FWDGT_PSC_DIV16</i>	FWDGT prescaler set to 16
<i>FWDGT_PSC_DIV32</i>	FWDGT prescaler set to 32
<i>FWDGT_PSC_DIV64</i>	FWDGT prescaler set to 64
<i>FWDGT_PSC_DIV128</i>	FWDGT prescaler set to 128
<i>FWDGT_PSC_DIV256</i>	FWDGT prescaler set to 256
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* configure 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 3-274. Function fwdgt_flag_get

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
<i>FWDGT_FLAG_PUD</i>	a write operation to FWDGT_PSC register is on going
<i>FWDGT_FLAG_RUD</i>	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);
```

3.12. GPIO

GPIO is used to implement logic input/output functions for the devices. The GPIO registers are listed in chapter [3.12.1](#), the GPIO firmware functions are introduced in chapter [3.12.2](#).

3.12.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

Table 3-275. GPIO Registers

Registers	Descriptions
GPIOx_CTL	GPIO port control register
GPIOx_OMODE	GPIO port output mode register
GPIOx_OSPD	GPIO port output speed register
GPIOx_PUD	GPIO port pull-up/pull-down register
GPIOx_ISTAT	GPIO port input status register
GPIOx_OCTL	GPIO port output control register
GPIOx_BOP	GPIO port bit operation register
GPIOx_LOCK	GPIO port configuration lock register
GPIOx_AFSEL0	GPIO alternate function selected register 0
GPIOx_AFSEL1	GPIO alternate function selected register 1
GPIOx_BC	GPIO bit clear register
GPIOx_TG	GPIO port bit toggle register
GPIOx_SCFG	GPIO secure configuration register

3.12.2. Descriptions of Peripheral functions

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

Table 3-276. GPIO firmware function

Function name	Function description
gpio_deinit	reset GPIO port
gpio_mode_set	set GPIO mode
gpio_output_options_set	set GPIO output type and speed
gpio_bit_set	set GPIO pin bit
gpio_bit_reset	reset GPIO pin bit
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_af_set	set GPIO alternate function

Function name	Function description
gpio_pin_lock	lock GPIO pin bit
gpio_bit_toggle	toggle GPIO pin status
gpio_port_toggle	toggle GPIO port status
gpio_bit_set_sec_cfg	configure GPIO pin bit secure configuration bit status to set
gpio_bit_reset_sec_cfg	configure GPIO pin bit secure configuration bit status to reset
gpio_sec_cfg_bit_get	get GPIO pin secure configuration bit status

gpio_deinit

The description of gpio_deinit is shown as below:

Table 3-277. 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)
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

gpio_mode_set

The description of gpio_mode_set is shown as below:

Table 3-278. Function gpio_mode_set

Function name	gpio_mode_set
Function prototype	void gpio_mode_set(uint32_t gpio_periph, uint32_t mode, uint32_t pull_up_down, uint32_t pin);
Function descriptions	set GPIO mode
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C)

Input parameter{in}	
mode	gpio pin mode
<i>GPIO_MODE_INPUT</i>	input mode
<i>GPIO_MODE_OUTPUT</i> <i>T</i>	output mode
<i>GPIO_MODE_AF</i>	alternate function mode
<i>GPIO_MODE_ANALOG</i> <i>G</i>	analog mode
Input parameter{in}	
pull_up_down	gpio pin with pull-up or pull-down resistor
<i>GPIO_PUPD_NONE</i>	floating mode, no pull-up and pull-down resistors
<i>GPIO_PUPD_PULLUP</i>	with pull-up resistor
<i>GPIO_PUPD_PULLDOWN</i> <i>WN</i>	with pull-down resistor
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure PA0 as input mode with pullup */
```

```
gpio_mode_set(GPIOA, GPIO_MODE_INPUT, GPIO_PUPD_PULLUP, GPIO_PIN_0);
```

gpio_output_options_set

The description of gpio_output_options_set is shown as below:

Table 3-279. Function gpio_output_options_set

Function name	gpio_output_options_set
Function prototype	void gpio_output_options_set(uint32_t gpio_periph, uint8_t otype, uint32_t speed, uint32_t pin);
Function descriptions	set GPIO output type and speed
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
otype	gpio pin output mode

<code>GPIO_OTYPE_PP</code>	push pull mode
<code>GPIO_OTYPE_OD</code>	open drain mode
Input parameter{in}	
speed	gpio pin output max speed
<code>GPIO_OSPEED_2MHZ</code>	output max speed 2MHz
<code>GPIO_OSPEED_10MHZ</code>	output max speed 10MHz
<code>GPIO_OSPEED_25MHZ</code>	output max speed 25MHz
<code>GPIO_OSPEED_166MHZ</code>	output max speed 166MHz
Input parameter{in}	
pin	GPIO pin
<code>GPIO_PIN_x</code>	GPIO_PIN_x (x=0..15)
<code>GPIO_PIN_ALL</code>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure PA0 as push pull mode */
```

```
gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_2MHZ,
GPIO_PIN_0);
```

gpio_bit_set

The description of gpio_bit_set is shown as below:

Table 3-280. 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 bit
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<code>GPIOx</code>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<code>GPIO_PIN_x</code>	GPIO_PIN_x(x=0..15)
<code>GPIO_PIN_ALL</code>	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 3-281. 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 bit
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset PA0 */
```

```
gpio_bit_reset(GPIOA, GPIO_PIN_0);
```

gpio_bit_write

The description of gpio_bit_write is shown as below:

Table 3-282. 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
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Input parameter{in}	
bit_value	SET or RESET
<i>RESET</i>	clear the port pin
<i>SET</i>	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 3-283. 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
<i>GPIOx</i>	GPIOx(x = A,B,C)
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 1010 0101 to Port A */
```

```
gpio_port_write(GPIOA, 0xA5A5);
```

gpio_input_bit_get

The description of gpio_input_bit_get is shown as below:

Table 3-284. 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
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get status of PA0 */
```

```
FlagStatus 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 3-285. 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 all pins input status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Output parameter{out}	
-	-
Return value	
uint16_t	0x0000-0xFFFF

Example:

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

uint16_t port_state;

port_state = gpio_input_port_get(GPIOA);
```

gpio_output_bit_get

The description of gpio_output_bit_get is shown as below:

Table 3-286. 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
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	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 3-287. 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 all pins output status
Precondition	-
The called functions	-

Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Output parameter{out}	
-	-
Return value	
uint16_t	0x0000-0xFFFF

Example:

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

```
uint16_t port_state;
```

```
port_state = gpio_output_port_get(GPIOA);
```

gpio_af_set

The description of gpio_af_set is shown as below:

Table 3-288. Function gpio_af_set

Function name	gpio_af_set
Function prototype	void gpio_af_set(uint32_t gpio_periph, uint32_t alt_func_num, uint32_t pin);
Function descriptions	set GPIO alternate function
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
alt_func_num	GPIO pin af function, please refer to specific device datasheet
<i>GPIO_AF_0</i>	USART0, USART1, TIMER0, SPI0, SPI1, CK_OUT0, RTC_REF, IR_OUT
<i>GPIO_AF_1</i>	USART1, TIMER0, TIMER1, TIMER2, I2S1
<i>GPIO_AF_2</i>	USART0, TIMER0, TIMER2, TIMER3, TIMER4, SPI0, I2S1, SDIO
<i>GPIO_AF_3</i>	QSPI, USART1, TIMER0, TIMER2, TIMER3, SQPI, TSI
<i>GPIO_AF_4</i>	TSI, TIMER4, SPI0, I2C0, QSPI, I2C1, I2S1
<i>GPIO_AF_5</i>	SPI0, SPI1, I2C0, I2S1
<i>GPIO_AF_6</i>	SPI0, SPI1, I2S1, I2C1
<i>GPIO_AF_7</i>	USART0, USART1, USART2, TIMER4, TIMER16, SPI0, SPI1, DCI(DCI not support on GD32W515Tx series devices)
<i>GPIO_AF_8</i>	USART2, SQPI, TIMER0, TIMER15
<i>GPIO_AF_9</i>	RTC, TIMER1, IR_OUT, I2C1
<i>GPIO_AF_10</i>	USART2, TIMER16, USBFS
<i>GPIO_AF_11</i>	TIMER15
<i>GPIO_AF_12</i>	SDIO, DCI(DCI not support on GD32W515Tx series devices)
<i>GPIO_AF_13</i>	DCI(DCI not support on GD32W515Tx series devices)

<i>GPIO_AF_14</i>	HPDF,DCI(HPDF and DCI not support on GD32W515Tx series devices)
<i>GPIO_AF_15</i>	EVENTOUT
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*set PA0 alternate function 0 */
gpio_af_set(GPIOA, GPIO_AF_0, GPIO_PIN_0);
```

gpio_pin_lock

The description of gpio_pin_lock is shown as below:

Table 3-289. 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 bit
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

gpio_bit_toggle

The description of gpio_bit_toggle is shown as below:

Table 3-290. Function gpio_bit_toggle

Function name	gpio_bit_toggle
Function prototype	void gpio_bit_toggle(uint32_t gpio_periph, uint32_t pin);
Function descriptions	toggle GPIO pin status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	GPIO_PIN_ALL
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* toggle PA0 */
```

```
gpio_bit_toggle(GPIOA, GPIO_PIN_0);
```

gpio_port_toggle

The description of gpio_port_toggle is shown as below:

Table 3-291. Function gpio_port_toggle

Function name	gpio_port_toggle
Function prototype	void gpio_port_toggle(uint32_t gpio_periph);
Function descriptions	toggle GPIO port status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* toggle GPIOA */

gpio_port_toggle(GPIOA);
```

gpio_bit_set_sec_cfg

The description of gpio_bit_set_sec_cfg is shown as below:

Table 3-292. Function gpio_bit_set_sec_cfg

Function name	gpio_bit_set_sec_cfg
Function prototype	void gpio_bit_set_sec_cfg(uint32_t gpio_periph, uint32_t pin);
Function descriptions	configure GPIO pin bit secure configuration bit status to set
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	GPIO_PIN_ALL
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPIOA pin 0 secure configuration bit status to set */

gpio_bit_set_sec_cfg(GPIOA, GPIO_PIN_0);
```

gpio_bit_reset_sec_cfg

The description of gpio_bit_reset_sec_cfg is shown as below:

Table 3-293. Function gpio_bit_reset_sec_cfg

Function name	gpio_bit_reset_sec_cfg
Function prototype	void gpio_bit_reset_sec_cfg(uint32_t gpio_periph, uint32_t pin);
Function descriptions	configure GPIO pin bit secure configuration bit status to reset
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)

Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	GPIO_PIN_ALL
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPIOA pin 0 secure configuration bit status to reset */
```

```
gpio_bit_reset_sec_cfg(GPIOA, GPIO_PIN_0);
```

gpio_sec_cfg_bit_get

The description of gpio_sec_cfg_bit_get is shown as below:

Table 3-294. Function gpio_sec_cfg_bit_get

Function name	gpio_sec_cfg_bit_get
Function prototype	FlagStatus gpio_sec_cfg_bit_get(uint32_t gpio_periph, uint32_t pin);
Function descriptions	get GPIO pin secure configuration bit status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	GPIO_PIN_ALL
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get GPIOA pin0 secure configuration bit status */
```

```
FlagStatus sec_cfg_bit_state = gpio_sec_cfg_bit_get(GPIOA, GPIO_PIN_0);
```

3.13. HAU

The HASH Acceleration Unit supports acceleration of SHA-1, SHA-224, SHA-256, MD5

algorithm and the HMAC (keyed-hash message authentication code) algorithm. The HAU registers are listed in chapter [3.13.1](#). the HAU firmware functions are introduced in chapter [3.13.2](#).

3.13.1. Descriptions of Peripheral registers

HAU registers are listed in the table shown as below:

Table 3-295. HAU Registers

Registers	Descriptions
HAU_CTL	control register
HAU_DI	data input register
HAU_CFG	configuration register
HAU_DO0	data output register 0
HAU_DO1	data output register 1
HAU_DO2	data output register 2
HAU_DO3	data output register 3
HAU_DO4	data output register 4
HAU_DO5	data output register 5
HAU_DO6	data output register 6
HAU_DO7	data output register 7
HAU_INTEN	interrupt enable register
HAU_STAT	status and interrupt flag register
HAU_CTXSx (x = 0..53)	context switch register

3.13.2. Descriptions of Peripheral functions

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

Table 3-296. HAU firmware function

Function name	Function description
hau_deinit	reset the HAU peripheral
hau_init	initialize the HAU peripheral parameters
hau_init_struct_para_init	initialize the struct hau_initpara
hau_reset	reset the HAU processor core
hau_last_word_validbits_num_config	configure the number of valid bits in last word of the message
hau_data_write	write data to the IN FIFO
hau_infifo_words_num_get	return the number of words already written into the IN FIFO
hau_digest_read	read the message digest result
hau_digest_calculation_enable	enable digest calculation
hau_multiple_single_dma_config	configure multiple or single DMA is used, and digest calculation at the end of a DMA transfer or not
hau_dma_enable	enable the HAU DMA interface

Function name	Function description
hau_dma_disable	disable the HAU DMA interface
hau_context_struct_para_init	initialize the struct context
hau_context_save	save the HAU peripheral context
hau_context_restore	restore the HAU peripheral context
hau_hash_sha_1	calculate digest using SHA1 in HASH mode
hau_hmac_sha_1	calculate digest using SHA1 in HMAC mode
hau_hash_sha_224	calculate digest using SHA224 in HASH mode
hau_hmac_sha_224	calculate digest using SHA224 in HMAC mode
hau_hash_sha_256	calculate digest using SHA256 in HASH mode
hau_hmac_sha_256	calculate digest using SHA256 in HMAC mode
hau_hash_md5	calculate digest using MD5 in HASH mode
hau_hmac_md5	calculate digest using MD5 in HMAC mode
hau_flag_get	get the HAU flag status
hau_flag_clear	clear the HAU flag status
hau_interrupt_enable	enable the HAU interrupts
hau_interrupt_disable	disable the HAU interrupts
hau_interrupt_flag_get	get the HAU interrupt flag status
hau_interrupt_flag_clear	clear the HAU interrupt flag status

Structure hau_init_parameter_struct

Table 3-297. Structure hau_init_parameter_struct

Member name	Function description
algo	algorithm selection: HAU_ALGO_SHA1, HAU_ALGO_SHA224, HAU_ALGO_SHA256, HAU_ALGO_MD5
mode	HAU mode selection: HAU_MODE_HASH, HAU_MODE_HMAC
datatype	data type mode: HAU_SWAPPING_32BIT, HAU_SWAPPING_16BIT, HAU_SWAPPING_8BIT, HAU_SWAPPING_1BIT
keytype	key length mode: HAU_KEY_SHORTER_64, HAU_KEY_LONGGER_64

Structure hau_digest_parameter_struct

Table 3-298. Structure hau_digest_parameter_struct

Member name	Function description
out[8]	message digest result 0-7

Structure hau_context_parameter_struct

Table 3-299. Structure hau_context_parameter_struct

Member name	Function description
hau_ctl_bak	backup of HAU_CTL register

hau_cfg_bak	backup of HAU_CFG register
hau_inten_bak	backup of HAU_INTEN register
hau_ctxs_bak[54]	backup of HAU_CTXSx registers

hau_deinit

The description of hau_deinit is shown as below:

Table 3-300. Function hau_deinit

Function name	hau_deinit
Function prototype	void hau_deinit(void);
Function descriptions	reset the HAU peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the HAU peripheral */
```

```
hau_deinit();
```

hau_init

The description of hau_init is shown as below:

Table 3-301. Function hau_init

Function name	hau_init
Function prototype	void hau_init(hau_init_parameter_struct* initpara);
Function descriptions	initialize the HAU peripheral parameters
Precondition	-
The called functions	-
Input parameter{in}	
initpara	the HAU peripheral parameters, refer to structure Table 3-297. Structure hau_init_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the HAU peripheral parameters */

hau_init_parameter_struct hau_initpara;

...

hau_init_struct_para_init(&hau_initpara);

hau_initpara.algo = algo;

hau_initpara.mode = HAU_MODE_HMAC;

hau_initpara.datatype = HAU_SWAPPING_8BIT;

if(key_len > 64U){

    hau_initpara.keytype = HAU_KEY_LONGGER_64;

}else{

    hau_initpara.keytype = HAU_KEY_SHORTER_64;

}

hau_init(&hau_initpara);

```

hau_init_struct_para_init

The description of hau_init_struct_para_init is shown as below:

Table 3-302. Function hau_init_struct_para_init

Function name	hau_init_struct_para_init
Function prototype	void hau_init_struct_para_init(hau_init_parameter_struct* initpara);
Function descriptions	initialize the sturct hau_initpara
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
initpara	the HAU peripheral parameters, refer to structure Table 3-297. Structure hau_init_parameter_struct
Return value	
-	-

Example:

```

/* initialize the HAU peripheral parameters */

hau_init_parameter_struct hau_initpara;

hau_init_struct_para_init(&hau_initpara);

```


hau_reset

The description of hau_reset is shown as below:

Table 3-303. Function hau_reset

Function name	hau_reset
Function prototype	void hau_reset(void);
Function descriptions	reset the HAU processor core
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the HAU processor core */
hau_reset();
```

hau_last_word_validbits_num_config

The description of hau_last_word_validbits_num_config is shown as below:

Table 3-304. Function hau_last_word_validbits_num_config

Function name	hau_last_word_validbits_num_config
Function prototype	void hau_last_word_validbits_num_config(uint32_t valid_num);
Function descriptions	configure the number of valid bits in last word of the message
Precondition	-
The called functions	-
Input parameter{in}	
valid_num	number of valid bits in last word of the message(0x00 – 0x1F)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the number of valid bits in last word of the message */
hau_last_word_validbits_num_config(0x10);
```

hau_data_write

The description of hau_data_write is shown as below:

Table 3-305. Function hau_data_write

Function name	hau_data_write
Function prototype	void hau_data_write(uint32_t data);
Function descriptions	write data to the IN FIFO
Precondition	-
The called functions	-
Input parameter{in}	
data	data to write (0x0 – 0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write data to the IN FIFO */
```

```
hau_data_write(0x10);
```

hau_infifo_words_num_get

The description of hau_infifo_words_num_get is shown as below:

Table 3-306. Function hau_infifo_words_num_get

Function name	hau_infifo_words_num_get
Function prototype	uint32_t hau_infifo_words_num_get(void);
Function descriptions	return the number of words already written into the IN FIFO
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	0x0 – 0xFFFFFFFF

Example:

```
/* return the number of words already written into the IN FIFO */
```

```
uint32_t num;
```

```
num = hau_infifo_words_num_get();
```

hau_digest_read

The description of hau_digest_read is shown as below:

Table 3-307. Function hau_digest_read

Function name	hau_digest_read
Function prototype	void hau_digest_read(hau_digest_parameter_struct* digestpara);
Function descriptions	read the message digest result
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
digestpara	the message digest result, refer to structure Table 3-298. Structure hau_digest_parameter_struct
Return value	
-	-

Example:

```
/* read the message digest result */
hau_digest_parameter_struct digestpara;
hau_digest_read(&digestpara);
```

hau_digest_calculation_enable

The description of hau_digest_calculation_enable is shown as below:

Table 3-308. Function hau_digest_calculation_enable

Function name	hau_digest_calculation_enable
Function prototype	void hau_digest_calculation_enable(void);
Function descriptions	enable digest calculation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable digest calculation */
hau_digest_calculation_enable();
```

hau_multiple_single_dma_config

The description of hau_multiple_single_dma_config is shown as below:

Table 3-309. Function hau_multiple_single_dma_config

Function name	hau_multiple_single_dma_config
Function prototype	void hau_multiple_single_dma_config(uint32_t multi_single);
Function descriptions	configure single or multiple DMA is used, and digest calculation at the end of a DMA transfer or not
Precondition	-
The called functions	-
Input parameter{in}	
multi_single	Multiple or single
<i>SINGLE_DMA_AUTO_DIGEST</i>	message padding and message digest calculation at the end of a DMA transfer
<i>MULTIPLE_DMA_NO_DIGEST</i>	multiple DMA transfers needed and CALEN bit is not automatically set at the end of a DMA transfer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure multiple or single DMA is used, and digest calculation at the end of a DMA transfer or not */
```

```
hau_multiple_single_dma_config(SINGLE_DMA_AUTO_DIGEST);
```

hau_dma_enable

The description of hau_dma_enable is shown as below:

Table 3-310. Function hau_dma_enable

Function name	hau_dma_enable
Function prototype	void hau_dma_enable(void);
Function descriptions	enable the HAU DMA interface
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the HAU DMA interface */
```

```
hau_dma_enable();
```

hau_dma_disable

The description of hau_dma_disable is shown as below:

Table 3-311. Function hau_dma_disable

Function name	hau_dma_disable
Function prototype	void hau_dma_disable(void);
Function descriptions	disable the HAU DMA interface
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the HAU DMA interface */
```

```
hau_dma_disable();
```

hau_context_struct_para_init

The description of hau_context_struct_para_init is shown as below:

Table 3-312. Function hau_context_struct_para_init

Function name	hau_context_struct_para_init
Function prototype	void hau_context_struct_para_init(hau_context_parameter_struct* context);
Function descriptions	initialize the struct context
Precondition	-
The called functions	-
Input parameter{in}	
context	the context, refer to structure Table 3-299. Structure hau_context_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the struct context */
```

```
hau_context_parameter_struct context_para;
```

```
hau_context_struct_para_init(&context_para);
```

hau_context_save

The description of hau_context_save is shown as below:

Table 3-313. Function hau_context_save

Function name	hau_context_save
Function prototype	void hau_context_save(hau_context_parameter_struct* context_save);
Function descriptions	save the HAU peripheral context
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
context_save	the HAU peripheral context, refer to structure Table 3-299. Structure hau_context_parameter_struct
Return value	
-	-

Example:

```
hau_context_parameter_struct context_para;
```

```
hau_context_struct_para_init(&context_para);
```

```
/* save HAU context */
```

```
hau_context_save(&context_para);
```

hau_context_restore

The description of hau_context_restore is shown as below:

Table 3-314. Function hau_context_restore

Function name	hau_context_restore
Function prototype	void hau_context_restore(hau_context_parameter_struct* context_restore);
Function descriptions	restore the HAU peripheral context
Precondition	-
The called functions	-
Input parameter{in}	
context_restore	the HAU peripheral context, refer to structure Table 3-299. Structure hau_context_parameter_struct
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
hau_context_parameter_struct context_para;
```

```
hau_context_struct_para_init(&context_para);
```

```
hau_context_save(&context_para);
```

```
.....
```

```
/* restore HAU context */
```

```
hau_context_restore(&context_para);
```

hau_hash_sha_1

The description of hau_hash_sha_1 is shown as below:

Table 3-315. Function hau_hash_sha_1

Function name	hau_hash_sha_1
Function prototype	ErrStatus hau_hash_sha_1(uint8_t *input, uint32_t in_length, uint8_t output[20]);
Function descriptions	calculate digest using SHA1 in HASH mode
Precondition	-
The called functions	-
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* calculate digest using SHA1 in HASH mode */
```

```
ErrStatus status = hau_hash_sha_1(&input, 0x10, output[0]);
```

hau_hmac_sha_1

The description of hau_hmac_sha_1 is shown as below:

Table 3-316. Function hau_hmac_sha_1

Function name	hau_hmac_sha_1
Function prototype	ErrStatus hau_hmac_sha_1(uint8_t *key, uint32_t keysize, uint8_t *input, uint32_t in_length, uint8_t output[20]);

Function descriptions	calculate digest using SHA1 in HMAC mode
Precondition	-
The called functions	-
Input parameter{in}	
key	pointer to the key used for HMAC
Input parameter{in}	
keysize	length of the key used for HMAC
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* calculate digest using SHA1 in HMAC mode */
```

```
ErrStatus status = hau_hmac_sha_1(&key, 0x10, &input, 0x10, output[0]);
```

hau_hash_sha_224

The description of hau_hash_sha_224 is shown as below:

Table 3-317. Function hau_hash_sha_224

Function name	hau_hash_sha_224
Function prototype	ErrStatus hau_hash_sha_224(uint8_t *input, uint32_t in_length, uint8_t output[28]);
Function descriptions	calculate digest using SHA224 in HASH mode
Precondition	-
The called functions	-
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* calculate digest using SHA224 in HASH mode */
```

```
ErrStatus status = hau_hash_sha_224 (&input, 0x10, output[0]);
```


hau_hmac_sha_224

The description of hau_hmac_sha_224 is shown as below:

Table 3-318. Function hau_hmac_sha_224

Function name	hau_hmac_sha_224
Function prototype	ErrStatus hau_hmac_sha_224(uint8_t *key, uint32_t keysize, uint8_t *input, uint32_t in_length, uint8_t output[28]);
Function descriptions	calculate digest using SHA224 in HMAC mode
Precondition	-
The called functions	-
Input parameter{in}	
key	pointer to the key used for HMAC
Input parameter{in}	
keysize	length of the key used for HMAC
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* calculate digest using SHA224 in HMAC mode */
```

```
ErrStatus status = hau_hmac_sha_224 (&key, 0x10, &input, 0x10, output[0]);
```

hau_hash_sha_256

The description of hau_hash_sha_256 is shown as below:

Table 3-319. Function hau_hash_sha_256

Function name	hau_hash_sha_256
Function prototype	ErrStatus hau_hash_sha_256(uint8_t *input, uint32_t in_length, uint8_t output[32]);
Function descriptions	calculate digest using SHA256 in HASH mode
Precondition	-
The called functions	-
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	

output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* calculate digest using SHA256 in HASH mode */
```

```
ErrStatus status = hau_hash_sha_256 (&input, 0x10, output[0]);
```

hau_hmac_sha_256

The description of hau_hmac_sha_256 is shown as below:

Table 3-320. Function hau_hmac_sha_256

Function name	hau_hmac_sha_256
Function prototype	ErrStatus hau_hmac_sha_256(uint8_t *key, uint32_t keysize, uint8_t *input, uint32_t in_length, uint8_t output[32]);
Function descriptions	calculate digest using SHA256 in HMAC mode
Precondition	-
The called functions	-
Input parameter{in}	
key	pointer to the key used for HMAC
Input parameter{in}	
keysize	length of the key used for HMAC
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* calculate digest using SHA256 in HMAC mode */
```

```
ErrStatus status = hau_hmac_sha_256 (&key, 0x10, &input, 0x10, output[0]);
```

hau_hash_md5

The description of hau_hash_md5 is shown as below:

Table 3-321. Function hau_hash_md5

Function name	hau_hash_md5
Function prototype	ErrStatus hau_hash_md5(uint8_t *input, uint32_t in_length, uint8_t output[16]);

Function descriptions	calculate digest using MD5 in HASH mode
Precondition	-
The called functions	-
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* calculate digest using MD5 in HASH mode */
```

```
ErrStatus status = hau_hash_md5 (&input, 0x10, output[0]);
```

hau_hmac_md5

The description of hau_hmac_md5 is shown as below:

Table 3-322. Function hau_hmac_md5

Function name	hau_hmac_md5
Function prototype	ErrStatus hau_hmac_md5(uint8_t *key, uint32_t keysize, uint8_t *input, uint32_t in_length, uint8_t output[16]);
Function descriptions	calculate digest using MD5 in HMAC mode
Precondition	-
The called functions	-
Input parameter{in}	
key	pointer to the key used for HMAC
Input parameter{in}	
keysize	length of the key used for HMAC
Input parameter{in}	
input	pointer to the input buffer
Input parameter{in}	
in_length	length of the input buffer
Output parameter{out}	
output	the result digest
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* calculate digest using MD5 in HMAC mode */
```

```
ErrStatus status = hau_hmac_md5 (&key, 0x10, &input, 0x10, output[0]);
```

hau_flag_get

The description of hau_flag_get is shown as below:

Table 3-323. Function hau_flag_get

Function name	hau_flag_get
Function prototype	FlagStatus hau_flag_get(uint32_t flag);
Function descriptions	get the HAU flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	HAU flag status
HAU_FLAG_DATA_INP UT	there is enough space (16 bytes) in the input FIFO
HAU_FLAG_CALCULA TION_COMPLETE	digest calculation is completed
HAU_FLAG_DMA	DMA is enabled (DMAE =1) or a transfer is processing
HAU_FLAG_BUSY	data block is in process
HAU_FLAG_INFIFO_N O_EMPTY	the input FIFO is not empty
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the HAU flag status */
```

```
FlagStatus status;
```

```
status = hau_flag_get (HAU_FLAG_DMA);
```

hau_flag_clear

The description of hau_flag_clear is shown as below:

Table 3-324. Function hau_flag_clear

Function name	hau_flag_clear
Function prototype	void hau_flag_clear(uint32_t flag);
Function descriptions	clear the HAU flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	HAU flag status
HAU_FLAG_DATA_INP	there is enough space (16 bytes) in the input FIFO

<i>UT</i>	
<i>HAU_FLAG_CALCULATION_COMPLETE</i>	digest calculation is completed
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the HAU flag status */
```

```
hau_flag_clear (HAU_FLAG_DATA_INPUT);
```

hau_interrupt_enable

The description of hau_interrupt_enable is shown as below:

Table 3-325. Function hau_interrupt_enable

Function name	hau_interrupt_enable
Function prototype	void hau_interrupt_enable(uint32_t interrupt);
Function descriptions	enable the HAU interrupts
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	HAU flag status
<i>HAU_INT_DATA_INPUT</i>	a new block can be entered into the IN buffer
<i>HAU_INT_CALCULATION_COMPLETE</i>	calculation complete
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable hau interrupt */
```

```
hau_interrupt_enable(HAU_INT_DATA_INPUT);
```

hau_interrupt_disable

The description of hau_interrupt_disable is shown as below:

Table 3-326. Function hau_interrupt_disable

Function name	hau_interrupt_disable
Function prototype	void hau_interrupt_disable(uint32_t interrupt);

Function descriptions	disable the HAU interrupts
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	HAU flag status
<i>HAU_INT_DATA_INPUT_T</i>	a new block can be entered into the IN buffer
<i>HAU_INT_CALCULATION_COMPLETE</i>	calculation complete
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable hau interrupt */
hau_interrupt_disable(HAU_INT_DATA_INPUT);
```

hau_interrupt_flag_get

The description of hau_interrupt_flag_get is shown as below:

Table 3-327. Function hau_interrupt_flag_get

Function name	hau_interrupt_flag_get
Function prototype	FlagStatus hau_interrupt_flag_get(uint32_t int_flag);
Function descriptions	get the HAU interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	HAU interrupt flag status
<i>HAU_INT_FLAG_DATA_INPUT</i>	there is enough space (16 bytes) in the input FIFO
<i>HAU_INT_FLAG_CALCULATION_COMPLETE</i>	digest calculation is completed
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the HAU interrupt flag status */
FlagStatus status = hau_interrupt_flag_get(HAU_INT_FLAG_DATA_INPUT);
```

hau_interrupt_flag_clear

The description of hau_interrupt_flag_clear is shown as below:

Table 3-328. Function hau_interrupt_flag_clear

Function name	hau_interrupt_flag_clear
Function prototype	void hau_interrupt_flag_clear(uint32_t int_flag)
Function descriptions	clear the HAU interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	HAU interrupt flag status
<i>HAU_INT_FLAG_DATA_INPUT</i>	there is enough space (16 bytes) in the input FIFO
<i>HAU_INT_FLAG_CALCULATION_COMPLETE</i>	digest calculation is completed
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the HAU interrupt flag status */
```

```
hau_interrupt_flag_clear(HAU_INT_FLAG_DATA_INPUT);
```

3.14. HPDF

A high performance digital filter module (HPDF) for external sigma delta (Σ - Δ) modulator is integrated in GD32W51x. The HPDF registers are listed in chapter [3.14.1](#), the HPDF firmware functions are introduced in chapter [3.14.2](#).

3.14.1. Descriptions of Peripheral registers

HPDF registers are listed in the table shown as below:

Table 3-329. HPDF Registers

Registers	Descriptions
HPDF_CHxCTL	HPDF Channel x control register 0
HPDF_CHxCFG0	HPDF Channel x configuration register 0
HPDF_CHxCFG1	HPDF Channel x configuration register 1
HPDF_CHxTMFDT	HPDF Channel x threshold monitor filter data register
HPDF_CHxPDI	HPDF Channel x input data register
HPDF_CHxPS	HPDF Channel x pulse skip register
HPDF_FLTyCTL0	HPDF Filter y control register 0
HPDF_FLTyCTL1	HPDF Filter y control register 1
HPDF_FLTySTAT	HPDF Filter y status register
HPDF_FLTyINTC	HPDF Filter y interrupt flag clear register
HPDF_FLTyIGCS	HPDF Filter y inserted group channel selection register
HPDF_FLTySFCFG	HPDF Filter y sinc filter configuration register
HPDF_FLTyIDATA	HPDF Filter y inserted group conversion data register
HPDF_FLTyRDATA	HPDF Filter y regular channel conversion data register
HPDF_FLTyTMHT	HPDF Filter y threshold monitor high threshold register
HPDF_FLTyTMLT	HPDF Filter y threshold monitor low threshold register
HPDF_FLTyTMSTAT	HPDF Filter y threshold monitor status register
HPDF_FLTyTMFC	HPDF Filter y threshold monitor flag clear register
HPDF_FLTyEMMAX	HPDF Filter y extremes monitor maximum register
HPDF_FLTyEMMIN	HPDF Filter y extremes monitor minimum register

3.14.2. Descriptions of Peripheral functions

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

Table 3-330. HPDF firmware function

Function name	Function description
hpdf_deinit	reset HPDF
hpdf_channel_struct_para_init	initialize the parameters of HPDF channel struct with the default values
hpdf_filter_struct_para_init	initialize the parameters of HPDF filter struct with the

Function name	Function description
	default values
hpdf_rc_struct_para_init	initialize the parameters of regular conversion struct with the default values
hpdf_ic_struct_para_init	initialize the parameters of inserted conversion struct with the default values
hpdf_enable	enable the HPDF module globally
hpdf_disable	disable the HPDF module globally
hpdf_channel_init	initialize the HPDF channel
hpdf_filter_init	initialize the HPDF filter
hpdf_rc_init	initialize the regular conversion
hpdf_ic_init	initialize the inserted conversion
hpdf_clock_output_config	configure serial output clock
hpdf_clock_output_source_config	configure serial clock output source
hpdf_clock_output_duty_mode_disable	disable serial clock output duty mode
hpdf_clock_output_duty_mode_enable	enable serial clock output duty mode
hpdf_clock_output_divider_config	configure serial clock output divider
hpdf_channel_enable	enable channel
hpdf_channel_disable	disable channel
hpdf_spi_clock_source_config	configure SPI clock source
hpdf_serial_interface_type_config	configure serial interface type
hpdf_malfunction_monitor_disable	disable malfunction monitor
hpdf_malfunction_monitor_enable	enable malfunction monitor
hpdf_clock_loss_disable	disable clock loss detector
hpdf_clock_loss_enable	enable clock loss detector
hpdf_channel_pin_redirection_disable	disable channel inputs pins redirection
hpdf_channel_pin_redirection_enable	enable channel inputs pins redirection
hpdf_channel_mux_config	configure channel multiplexer select input data source
hpdf_data_pack_mode_config	configure data packing mode
hpdf_data_right_bit_shift_config	configure data right bit-shift
hpdf_calibration_offset_config	configure calibration offset
hpdf_malfunction_break_signal_config	configure malfunction monitor break signal
hpdf_malfunction_counter_config	configure malfunction monitor counter threshold
hpdf_write_parallel_data_standard_mode	write the parallel data on standard mode of data packing
hpdf_write_parallel_data_interleaved_mode	write the parallel data on interleaved mode of data packing
hpdf_write_parallel_data_dual_mode	write the parallel data on dual mode of data packing
hpdf_pulse_skip_update	update the number of pulses to skip
hpdf_pulse_skip_read	read the number of pulses to skip
hpdf_filter_enable	enable filter
hpdf_filter_disable	disable filter
hpdf_filter_config	configure sinc filter order and oversample

Function name	Function description
hpdf_integrator_oversample	configure integrator oversampling rate
hpdf_threshold_monitor_filter_config	configure threshold monitor filter order and oversample
hpdf_threshold_monitor_filter_read_data	read the threshold monitor filter data
hpdf_threshold_monitor_fast_mode_disable	disable threshold monitor fast mode
hpdf_threshold_monitor_fast_mode_enable	enable threshold monitor fast mode
hpdf_threshold_monitor_channel	configure threshold monitor channel
hpdf_threshold_monitor_high_threshold	configure threshold monitor high threshold value
hpdf_threshold_monitor_low_threshold	configure threshold monitor low threshold value
hpdf_high_threshold_break_signal	configure threshold monitor high threshold event break signal
hpdf_low_threshold_break_signal	configure threshold monitor low threshold event break signal
hpdf_extremes_monitor_channel	configure extremes monitor channel
hpdf_extremes_monitor_maximum_get	get the extremes monitor maximum value
hpdf_extremes_monitor_minimum_get	get the extremes monitor minimum value
hpdf_rc_continuous_disable	disable regular conversions continuous mode
hpdf_rc_continuous_enable	enable regular conversions continuous mode
hpdf_rc_start_by_software	start regular channel conversion by software
hpdf_rc_syn_disable	disable regular conversion synchronously
hpdf_rc_syn_enable	enable regular conversion synchronously
hpdf_rc_dma_disable	disable regular conversion DMA channel
hpdf_rc_dma_enable	enable regular conversion DMA channel
hpdf_rc_channel_config	configure regular conversion channel
hpdf_rc_fast_mode_disable	disable regular conversion fast conversion mode
hpdf_rc_fast_mode_enable	enable regular conversion fast conversion mode
hpdf_rc_data_get	get the regular conversion data
hpdf_rc_channel_get	get the channel of regular channel most recently converted
hpdf_ic_start_by_software	start inserted channel conversion by software
hpdf_ic_syn_disable	disable inserted conversion synchronously
hpdf_ic_syn_enable	enable inserted conversion synchronously
hpdf_ic_dma_disable	disable inserted conversion DMA channel
hpdf_ic_dma_enable	enable inserted conversion DMA channel
hpdf_ic_scan_mode_disable	disable scan conversion mode
hpdf_ic_scan_mode_enable	enable scan conversion mode
hpdf_ic_trigger_signal_disable	disable inserted conversions trigger signal
hpdf_ic_trigger_signal_config	configure inserted conversions trigger signal and trigger edge
hpdf_ic_channel_config	configure inserted group conversions channel

Function name	Function description
hpdf_ic_data_get	get the inserted conversions data
hpdf_ic_channel_get	get the channel of inserted group channel most recently converted
hpdf_flag_get	get the HPDF flags
hpdf_flag_clear	clear the HPDF flags
hpdf_interrupt_enable	enable HPDF interrupt
hpdf_interrupt_disable	disable HPDF interrupt
hpdf_interrupt_flag_get	get the HPDF interrupt flags
hpdf_interrupt_flag_clear	clear the HPDF interrupt flags

Enum hpdf_channel_enum

Table 3-331. Enum hpdf_channel_enum

enum name	enum description
CHANNEL0	HPDF channel0
CHANNEL1	HPDF channel1

Enum hpdf_filter_enum

Table 3-332. Enum hpdf_filter_enum

enum name	enum description
FLT0	HPDF filter0
FLT0	HPDF filter1

Enum hpdf_flag_enum

Table 3-333. Enum hpdf_flag_enum

enum name	enum description
HPDF_FLAG_FLTy_ICEF	inserted conversion end flag
HPDF_FLAG_FLTy_RCEF	regular conversion end flag
HPDF_FLAG_FLTy_ICDOF	inserted conversion overflow flag
HPDF_FLAG_FLTy_RCDOF	regular conversion overflow flag
HPDF_FLAG_FLTy_TMEOF	threshold monitor event occurred flag
HPDF_FLAG_FLTy_ICPF	inserted conversion in progress flag
HPDF_FLAG_FLTy_RCPF	regular conversion in progress flag
HPDF_FLAG_FLT0	clock loss on channel 0 flag

enum name	enum description
_CKLF0	
HPDF_FLAG_FLT0 _CKLF1	clock loss on channel 1 flag
HPDF_FLAG_FLT0 _MMF0	malfunction event occurred on channel 0 flag
HPDF_FLAG_FLT0 _MMF1	malfunction event occurred on channel 1 flag
HPDF_FLAG_FLTy _RCHPDT	regular channel pending data
HPDF_FLAG_FLTy _LTF0	threshold monitor low threshold on channel 0 flag
HPDF_FLAG_FLTy _LTF1	threshold monitor low threshold on channel 1 flag
HPDF_FLAG_FLTy _HTF0	threshold monitor high threshold on channel 0 flag
HPDF_FLAG_FLTy _HTF1	threshold monitor high threshold on channel 1 flag

Enum hpdf_interrput_flag_enum

Table 3-334. Enum hpdf_interrput_flag_enum

enum name	enum description
HPDF_INT_FLAG_ FLTy_ICEF	inserted conversion end interrupt flag
HPDF_INT_FLAG_ FLTy_RCEF	regular conversion end interruptflag
HPDF_INT_FLAG_ FLTy_ICDOF	inserted conversion overflow interrupt flag
HPDF_INT_FLAG_ FLTy_RCDOF	regular conversion overflow interrupt flag
HPDF_INT_FLAG_ FLTy_TMEOF	threshold monitor event occurred interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF0	clock loss on channel 0 interrupt flag
HPDF_INT_FLAG_ FLT0_CKLF1	clock loss on channel 1 interrupt flag
HPDF_INT_FLAG_ FLT0_MMF0	malfunction monitor detection on channel 0 interrupt flag
HPDF_INT_FLAG_ FLT0_MMF1	malfunction monitor detection on channel 1 interrupt flag

Enum `hpdf_interrput_enum`

Table 3-335. Enum `hpdf_interrput_enum`

enum name	enum description
HPDF_INT_FLTy_I CEIE	inserted conversion end interrupt enable
HPDF_INT_FLTy_R CEIE	regular conversion end interrupt enable
HPDF_INT_FLTy_I CDOIE	inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_R CDOIE	regular conversion data overflow interrupt enable
HPDF_INT_FLTy_T MIE	threshold monitor interrupt enable
HPDF_INT_FLT0_M MIE	malfunction monitor interrupt enable
HPDF_INT_FLT0_C KLIE	clock loss interrupt enable

Structure `hpdf_channel_parameter_struct`

Table 3-336. Structure `hpdf_channel_parameter_struct`

Member name	Function description
<code>data_packing_mode</code>	ata packing mode for HPDF_CHxPDI register
<code>channel_muxlexer</code>	channel multiplexer select input data source
<code>channel_pin_select</code>	channel inputs pins selection
<code>ck_loss_detector</code>	clock loss detector
<code>malfunction_monitor</code>	malfunction monitor
<code>spi_ck_source</code>	SPI clock source select
<code>serial_interface</code>	serial interface type
<code>calibration_offset</code>	24-bit calibration offset
<code>right_bit_shift</code>	data right bit-shift
<code>tm_filter</code>	threshold monitor Sinc filter order selection
<code>tm_filter_oversample</code>	threshold monitor filter oversampling rate
<code>mm_break_signal</code>	malfunction monitor break signal distribution
<code>mm_counter_threshold</code>	malfunction monitor counter threshold
<code>plsk_value</code>	the number of serial input samples that will be skipped

Structure `hpdf_filter_parameter_struct`

Table 3-337. Structure `hpdf_filter_parameter_struct`

Member name	Function description
<code>tm_fast_mode</code>	threshold monitor fast mode
<code>tm_channel</code>	threshold monitor channel

tm_high_threshold	threshold monitor high threshold
tm_low_threshold	threshold monitor low threshold value
extreme_monitor_channel	extremes monitor channel
sinc_filter	sinc filter order
sinc_oversample	sinc filter oversampling rate
integrator_oversample	integrator oversampling rate
ht_break_signal	high threshold event break signal distribution
lt_break_signal	low threshold event break signal distribution

Structure hpdf_rc_parameter_struct

Table 3-338. Structure hpdf_rc_parameter_struct

Member name	Function description
fast_mode	fast conversion mode enable for regular conversions
rsc_channel	regular conversion channel
rcdmaen	DMA channel enabled to read data for the regular conversion
rcsyn	regular conversion synchronously
continuous_mode	regular conversions continuous mode

Structure hpdf_ic_parameter_struct

Table 3-339. Structure hpdf_ic_parameter_struct

Member name	Function description
trigger_dege	inserted conversions trigger edge
trigger_signal	inserted conversions trigger signal
icdmaen	DMA channel enabled to read data for the inserted channel group
scmod	scan conversion mode of inserted conversions
icsyn	inserted conversion synchronously
ic_channel_group	inserted channel group selection

hpdf_deinit

The description of hpdf_deinit is shown as below:

Table 3-340. Function hpdf_deinit

Function name	hpdf_deinit
Function prototype	void hpdf_deinit(void);
Function descriptions	reset HPDF
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* reset HPDF */
```

```
hpdf_deinit();
```

hpdf_channel_struct_para_init

The description of hpdf_channel_struct_para_init is shown as below:

Table 3-341. Function hpdf_channel_struct_para_init

Function name	hpdf_channel_struct_para_init
Function prototype	void hpdf_channel_struct_para_init(hpdf_channel_parameter_struct* init_struct);
Function descriptions	initialize the parameters of HPDF channel struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF channel, refer to Table 3-336. Structure hpdf_channel_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of HPDF channel */
```

```
hpdf_channel_parameter_struct hpdf_channel_init_struct;
```

```
hpdf_channel_struct_para_init(&hpdf_channel_init_struct);
```

hpdf_filter_struct_para_init

The description of hpdf_filter_struct_para_init is shown as below:

Table 3-342. Function hpdf_filter_struct_para_init

Function name	hpdf_filter_struct_para_init
Function prototype	void hpdf_filter_struct_para_init(hpdf_filter_parameter_struct* init_struct);
Function descriptions	initialize the parameters of HPDF filter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF filter, refer to Table 3-337.

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

Example:

```
/* initialize the parameters of HPDF filter */
hpdf_filter_parameter_struct hpdf_filter_init_struct;
hpdf_filter_struct_para_init(&hpdf_filter_init_struct);
```

hpdf_rc_struct_para_init

The description of hpdf_rc_struct_para_init is shown as below:

Table 3-343. Function hpdf_rc_struct_para_init

Function name	hpdf_rc_struct_para_init
Function prototype	void hpdf_rc_struct_para_init(hpdf_rc_parameter_struct* init_struct);
Function descriptions	initialize the parameters of regular conversion struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF regular conversion, refer to Table 3-338. Structure hpdf_rc_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of regular conversion */
hpdf_rc_parameter_struct hpdf_rc_init_struct;
hpdf_rc_struct_para_init(&hpdf_rc_init_struct);
```

hpdf_ic_struct_para_init

The description of hpdf_ic_struct_para_init is shown as below:

Table 3-344. Function hpdf_ic_struct_para_init

Function name	hpdf_ic_struct_para_init
Function prototype	void hpdf_ic_struct_para_init(hpdf_ic_parameter_struct* init_struct);
Function descriptions	initialize the parameters of inserted conversion struct with the default values
Precondition	-

The called functions	-
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF inserted conversion, refer to Table 3-339. Structure hpdf_ic_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of inserted conversion */
hpdf_ic_parameter_struct hpdf_ic_init_struct;
hpdf_ic_struct_para_init(&hpdf_ic_init_struct);
```

hpdf_enable

The description of hpdf_enable is shown as below:

Table 3-345. Function hpdf_enable

Function name	hpdf_enable
Function prototype	void hpdf_enable(void);
Function descriptions	enable the HPDF module globally
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the HPDF module */
hpdf_enable();
```

hpdf_disable

The description of hpdf_disable is shown as below:

Table 3-346. Function hpdf_disable

Function name	hpdf_disable
Function prototype	void hpdf_disable(void);
Function descriptions	disable the HPDF module globally

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

Example:

```
/* disable the HPDF module */
```

```
hpdf_disable();
```

hpdf_channel_init

The description of hpdf_channel_init is shown as below:

Table 3-347. Function hpdf_channel_init

Function name	hpdf_channel_init
Function prototype	void hpdf_channel_init(hpdf_channel_enum channelx, hpdf_channel_parameter_struct* init_struct);
Function descriptions	initialize the HPDF channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..1)	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF channel, refer to Table 3-336. Structure hpdf_channel_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the HPDF channel0 */
```

```
hpdf_channel_parameter_struct hpdf_channel_init_struct;
```

```
/* initialize HPDF channel0 */
```

```
hpdf_channel_init_struct.data_packing_mode = DPM_STANDARD_MODE;
```

```
hpdf_channel_init_struct.malfunction_monitor = MM_ENABLE;
```

```
hpdf_channel_init_struct.spi_ck_source = EXTERNAL_CKIN;
```

```
hpdf_channel_init_struct.channel_muxlexer = SERIAL_INPUT;
```

```
hpdf_channel_init_struct.serial_interface = SPI_RISING_EDGE;
```

```
hpdf_channel_init_struct.calibration_offset = 0;
```

```
hpdf_channel_init_struct.right_bit_shift = 0;
```

```
hpdf_channel_init_struct.mm_counter_threshold = 110;
```

```
hpdf_channel_init_struct.plsk_value = 0;
```

```
hpdf_channel_init(CHANNEL0, &hpdf_channel_init_struct);
```

hpdf_filter_init

The description of hpdf_filter_init is shown as below:

Table 3-348. Function hpdf_filter_init

Function name	hpdf_filter_init
Function prototype	void hpdf_filter_init(hpdf_filter_enum filtery, hpdf_filter_parameter_struct* init_struct);
Function descriptions	initialize the HPDF filter
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF filter, refer to Table 3-337. Structure hpdf_filter_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the HPDF fliter0 */
```

```
hpdf_filter_parameter_struct hpdf_filter_init_struct;
```

```
hpdf_filter_init_struct.tm_fast_mode = TMFM_DISABLE;
```

```
hpdf_filter_init_struct.tm_channel = TMCHEN_CHANNEL0;
```

```
hpdf_filter_init_struct.tm_high_threshold = tm_high_val;
```

```
hpdf_filter_init_struct.tm_low_threshold = tm_low_val;
```

```

hpdf_filter_init_struct.extreme_monitor_channel = EM_CHANNEL0;

hpdf_filter_init_struct.sinc_filter            = FLT_SINC3;

hpdf_filter_init_struct.sinc_oversample       = FLT_OVER_SAMPLE_32;

hpdf_filter_init_struct.integrator_oversample = INTEGRATOR_BYPASS;

hpdf_filter_init(FLT0, &hpdf_filter_init_struct);

```

hpdf_rc_init

The description of hpdf_rc_init is shown as below:

Table 3-349. Function hpdf_rc_init

Function name	hpdf_rc_init
Function prototype	void hpdf_rc_init(hpdf_filter_enum filtery, hpdf_rc_parameter_struct* init_struct);
Function descriptions	initialize the regular conversion
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF regular conversion, refer to Table 3-338. Structure hpdf_rc_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize regular conversion of the HPDF filter0 */

hpdf_rc_parameter_struct hpdf_rc_init_struct;

hpdf_rc_init_struct.fast_mode = FAST_DISABLE;

hpdf_rc_init_struct.rcs_channel = RCS_CHANNEL0;

hpdf_rc_init_struct.rcdmaen = RCDMAEN_ENABLE;

hpdf_rc_init_struct.continuous_mode = RCCM_ENABLE;

hpdf_rc_init(FLT0, &hpdf_rc_init_struct);

```

hpdf_ic_init

The description of hpdf_ic_init is shown as below:

Table 3-350. Function `hpdf_ic_init`

Function name	<code>hpdf_ic_init</code>
Function prototype	<code>void hpdf_ic_init(hpdf_filter_enum filtery, hpdf_ic_parameter_struct* init_struct);</code>
Function descriptions	initialize the inserted conversion
Precondition	-
The called functions	-
Input parameter{in}	
<code>filtery</code>	The filter of HPDF module
<code>FLTy(y=0..1)</code>	select HPDF filter, refer to Table 3-332. Enum <code>hpdf_filter_enum</code>
Input parameter{in}	
<code>*init_struct</code>	the initialized struct needed to initialize HPDF inserted conversion, refer to Table 3-339. Structure <code>hpdf_ic_parameter_struct</code>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize inserted conversion of the HPDF filter0 */
hpdf_ic_parameter_struct hpdf_ic_init_struct;
hpdf_ic_init_struct.ic_channel_group = IGCSEL_CHANNEL0_1;
hpdf_ic_init_struct.scmmod = SCMOD_ENABLE;
hpdf_ic_init_struct.icdmaen = ICDMAEN_ENABLE;
hpdf_ic_init_struct.icsyn = ICSYN_DISABLE ;
hpdf_ic_init(FLT0, &hpdf_ic_init_struct);

```

`hpdf_clock_output_config`

The description of `hpdf_clock_output_config` is shown as below:

Table 3-351. Function `hpdf_clock_output_config`

Function name	<code>hpdf_clock_output_config</code>
Function prototype	<code>void hpdf_clock_output_config(uint32_t source, uint8_t divider, uint32_t mode);</code>
Function descriptions	configure serial output clock
Precondition	disable HPDF
The called functions	-
Input parameter{in}	
<code>source</code>	the HPDF serial clock output source
<code>SERIAL_SYSTEM_CLK</code>	serial clock output source is from system clock

<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from audio clock
Input parameter{in}	
divider	serial clock output divider(0-255)
Input parameter{in}	
mode	serial clock output duty mode
<i>CKOUTDM_DISABLE</i>	disable serial clock output duty mode
<i>CKOUTDM_ENABLE</i>	enable serial clock output duty mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure serial clock output */
```

```
hpdf_clock_output_config(SERIAL_SYSTEM_CLK, 175, CKOUTDM_ENABLE);
```

hpdf_clock_output_source_config

The description of hpdf_clock_output_source_config is shown as below:

Table 3-352. Function hpdf_clock_output_source_config

Function name	hpdf_clock_output_source_config
Function prototype	void hpdf_clock_output_source_config(uint32_t source);
Function descriptions	configure serial clock output source
Precondition	disable HPDF
The called functions	-
Input parameter{in}	
source	the HPDF serial clock output source
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from system clock
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from audio clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure serial clock output source */
```

```
hpdf_clock_output_config(SERIAL_SYSTEM_CLK);
```

hpdf_clock_output_duty_mode_disable

The description of hpdf_clock_output_duty_mode_disable is shown as below:

Table 3-353. Function hpdf_clock_output_duty_mode_disable

Function name	hpdf_clock_output_duty_mode_disable
Function prototype	void hpdf_clock_output_duty_mode_disable(void);
Function descriptions	disable serial clock output duty mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable serial clock output duty mode*/
```

```
hpdf_clock_output_duty_mode_disable();
```

hpdf_clock_output_duty_mode_enable

The description of hpdf_clock_output_duty_mode_enable is shown as below:

Table 3-354. Function hpdf_clock_output_duty_mode_enable

Function name	hpdf_clock_output_duty_mode_enable
Function prototype	void hpdf_clock_output_duty_mode_enable(void);
Function descriptions	enable serial clock output duty mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable serial clock output duty mode*/
```

```
hpdf_clock_output_duty_mode_enable();
```

hpdf_clock_output_divider_config

The description of hpdf_clock_output_divider_config is shown as below:

Table 3-355. Function hpdf_clock_output_divider_config

Function name	hpdf_clock_output_divider_config
----------------------	----------------------------------

Function prototype	void hpdf_clock_output_divider_config(uint8_t divider);
Function descriptions	configure serial clock output divider
Precondition	disable HPDF
The called functions	-
Input parameter{in}	
divider	serial clock output divider(0-255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure serial clock output divider */
hpdf_clock_output_divider_config (255);
```

hpdf_channel_enable

The description of hpdf_channel_enable is shown as below:

Table 3-356. Function hpdf_channel_enable

Function name	hpdf_channel_enable
Function prototype	void hpdf_channel_enable(hpdf_channel_enum channelx);
Function descriptions	enable channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable channel0 */
hpdf_channel_enable(CHANNEL0);
```

hpdf_channel_disable

The description of hpdf_channel_disable is shown as below:

Table 3-357. Function hpdf_channel_disable

Function name	hpdf_channel_disable
Function prototype	void hpdf_channel_disable(hpdf_channel_enum channelx);

Function descriptions	disable channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable channel0 */
```

```
hpdf_channel_disable(CHANNEL0);
```

hpdf_spi_clock_source_config

The description of hpdf_spi_clock_source_config is shown as below:

Table 3-358. Function hpdf_spi_clock_source_config

Function name	hpdf_spi_clock_source_config
Function prototype	void hpdf_spi_clock_source_config(hpdf_channel_enum channelx, uint32_t clock_source);
Function descriptions	configure SPI clock source
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
clock_source	SPI clock source
<i>EXTERNAL_CKIN</i>	external input clock
<i>INTERNAL_CKOUT</i>	internal CKOUT clock
<i>HALF_CKOUT_FALLING_EDGE</i>	internal CKOUT clock, sampling point on each second CKOUT falling edge
<i>HALF_CKOUT_RISING_EDGE</i>	internal CKOUT clock, sampling point on each second CKOUT rising edge
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure external input clock as SPI clock source */
```

```
hpdf_spi_clock_source_config(CHANNEL0, EXTERNAL_CKIN);
```

hpdf_serial_interface_type_config

The description of hpdf_serial_interface_type_config is shown as below:

Table 3-359. Function hpdf_serial_interface_type_config

Function name	hpdf_serial_interface_type_config
Function prototype	void hpdf_serial_interface_type_config(hpdf_channel_enum channelx, uint32_t type);
Function descriptions	configure serial interface type
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..1)	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
type	serial interface type
SPI_RISING_EDGE	SPI interface, sample data on rising edge
SPI_FALLING_EDGE	SPI interface, sample data on falling edge
MANCHESTER_CODE 0	Manchester coded input: rising edge = logic 0, falling edge = logic 1
MANCHESTER_CODE 1	Manchester coded input: rising edge = logic 1, falling edge = logic 0
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure external input clock as SPI clock source */
```

```
hpdf_spi_clock_source_config(CHANNEL0, EXTERNAL_CKIN);
```

hpdf_malfunction_monitor_disable

The description of hpdf_malfunction_monitor_disable is shown as below:

Table 3-360. Function hpdf_malfunction_monitor_disable

Function name	hpdf_malfunction_monitor_disable
Function prototype	void hpdf_malfunction_monitor_disable(hpdf_channel_enum channelx);
Function descriptions	disable malfunction monitor
Precondition	-
The called functions	-

Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable malfunction monitor */
```

```
hpdf_malfunction_monitor_disable(CHANNEL0);
```

hpdf_malfunction_monitor_enable

The description of hpdf_malfunction_monitor_enable is shown as below:

Table 3-361. Function hpdf_malfunction_monitor_enable

Function name	hpdf_malfunction_monitor_enable
Function prototype	void hpdf_malfunction_monitor_enable(hpdf_channel_enum channelx);
Function descriptions	enable malfunction monitor
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable malfunction monitor */
```

```
hpdf_malfunction_monitor_enable(CHANNEL1);
```

hpdf_clock_loss_disable

The description of hpdf_clock_loss_disable is shown as below:

Table 3-362. Function hpdf_clock_loss_disable

Function name	hpdf_clock_loss_disable
Function prototype	void hpdf_clock_loss_disable(hpdf_channel_enum channelx);
Function descriptions	disable clock loss detector
Precondition	disable CHANNELx(x=0..1)
The called functions	-

Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable clock loss detector */
```

```
hpdf_clock_loss_disable(CHANNEL0);
```

hpdf_clock_loss_enable

The description of hpdf_clock_loss_enable is shown as below:

Table 3-363. Function hpdf_clock_loss_enable

Function name	hpdf_clock_loss_enable
Function prototype	void hpdf_clock_loss_enable(hpdf_channel_enum channelx);
Function descriptions	enable clock loss detector
Precondition	disable CHANNELx(x=0..1)
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable clock loss detector */
```

```
hpdf_clock_loss_enable(CHANNEL0);
```

hpdf_channel_pin_redirection_disable

The description of hpdf_channel_pin_redirection_disable is shown as below:

Table 3-364. Function hpdf_channel_pin_redirection_disable

Function name	hpdf_channel_pin_redirection_disable
Function prototype	void hpdf_channel_pin_redirection_disable(hpdf_channel_enum channelx);
Function descriptions	disable channel inputs pins redirection
Precondition	disable CHANNELx(x=0..1)
The called functions	-

Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable channel0 inputs pins redirection */
```

```
hpdf_channel_pin_redirection_disable(CHANNEL0);
```

hpdf_channel_pin_redirection_enable

The description of hpdf_channel_pin_redirection_enable is shown as below:

Table 3-365. Function hpdf_channel_pin_redirection_enable

Function name	hpdf_channel_pin_redirection_enable
Function prototype	void hpdf_channel_pin_redirection_enable(hpdf_channel_enum channelx);
Function descriptions	enable channel inputs pins redirection
Precondition	disable CHANNELx(x=0..1)
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable channel0 inputs pins redirection */
```

```
hpdf_channel_pin_redirection_enable(CHANNEL0);
```

hpdf_channel_multiplexer_config

The description of hpdf_channel_multiplexer_config is shown as below:

Table 3-366. Function hpdf_channel_multiplexer_config

Function name	hpdf_channel_multiplexer_config
Function prototype	void hpdf_channel_multiplexer_config(hpdf_channel_enum channelx, uint32_t data_source);
Function descriptions	configure channel multiplexer select input data source
Precondition	disable CHANNELx(x=0..1)

The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
data_source	input data source
<i>SERIAL_INPUT</i>	input data source is taken from serial inputs
<i>INTERNAL_INPUT</i>	input data source is taken from internal HPDF_CHxPDI register
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure channel multiplexer select input data source */
hpdf_channel_mux_config(CHANNEL0, SERIAL_INPUT);
```

hpdf_data_pack_mode_config

The description of hpdf_data_pack_mode_config is shown as below:

Table 3-367. Function hpdf_data_pack_mode_config

Function name	hpdf_data_pack_mode_config
Function prototype	void hpdf_data_pack_mode_config(hpdf_channel_enum channelx, uint32_t mode);
Function descriptions	configure data packing mode
Precondition	disable CHANNELx(x=0..1)
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
mode	parallel data packing mode
<i>DPM_STANDARD_MODE</i>	standard mode
<i>DPM_INTERLEAVED_MODE</i>	interleaved mode
<i>DPM_DUAL_MODE</i>	dual mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure data packing mode */
```

```
hpdf_data_pack_mode_config(CHANNEL0, DPM_STANDARD_MODE);
```

hpdf_data_right_bit_shift_config

The description of hpdf_data_right_bit_shift_config is shown as below:

Table 3-368. Function hpdf_data_right_bit_shift_config

Function name	hpdf_data_right_bit_shift_config
Function prototype	void hpdf_data_right_bit_shift_config(hpdf_channel_enum channelx, uint8_t right_shift);
Function descriptions	configure data right bit-shift
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..1)	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
right_shift	the number of bits that determine the right shift(0-31)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure data right bit-shift */
```

```
hpdf_data_right_bit_shift_config(CHANNEL0, 5);
```

hpdf_calibration_offset_config

The description of hpdf_calibration_offset_config is shown as below:

Table 3-369. Function hpdf_calibration_offset_config

Function name	hpdf_calibration_offset_config
Function prototype	void hpdf_calibration_offset_config(hpdf_channel_enum channelx, int32_t offset);
Function descriptions	configure calibration offset
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..1)	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
offset	24-bit calibration offset, must be in (-8388608~8388607)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure calibration offset */
```

```
hpdf_calibration_offset_config (CHANNEL0, -255);
```

hpdf_malfunction_break_signal_config

The description of hpdf_malfunction_break_signal_config is shown as below:

Table 3-370. Function hpdf_malfunction_break_signal_config

Function name	hpdf_malfunction_break_signal_config
Function prototype	void hpdf_malfunction_break_signal_config(hpdf_channel_enum channelx, uint32_t break_signal);
Function descriptions	configure malfunction monitor break signal
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..1)	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
break_signal	malfunction monitor break signal distribution
NO_MM_BREAK	break signal is not distributed to malfunction monitor on channel
MM_BREAK0	break signal 0 is distributed to malfunction monitor on channel
MM_BREAK1	break signal 1 is distributed to malfunction monitor on channel
MM_BREAK0_1	break signal 0 and 1 is distributed to malfunction monitor on channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure break signal is distributed to malfunction monitor on channel0 */
```

```
hpdf_data_pack_mode_config(CHANNEL0, MM_BREAK0_1);
```

hpdf_malfunction_counter_config

The description o hpdf_malfunction_counter_config is shown as below:

Table 3-371. Function hpdf_malfunction_counter_config

Function name	hpdf_malfunction_counter_config
---------------	---------------------------------

Function prototype	void hpdf_malfunction_counter_config(hpdf_channel_enum channelx, uint8_t threshold);
Function descriptions	configure malfunction monitor counter threshold
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
threshold	malfunction monitor counter threshold(0-255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure malfunction monitor counter threshold */
hpdf_malfunction_counter_config(CHANNEL0, 255);
```

hpdf_write_parallel_data_standard_mode

The description of hpdf_write_parallel_data_standard_mode is shown as below:

Table 3-372. Function hpdf_write_parallel_data_standard_mode

Function name	hpdf_write_parallel_data_standard_mode
Function prototype	void hpdf_write_parallel_data_standard_mode(hpdf_channel_enum channelx, int16_t data);
Function descriptions	write the parallel data on standard mode of data packing
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
data	the parallel data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write the parallel data on standard mode of data packing */
hpdf_write_parallel_data_standard_mode(CHANNEL0, 0xEFFF);
```

hpdf_write_parallel_data_interleaved_mode

The description of hpdf_write_parallel_data_interleaved_mode is shown as below:

Table 3-373. Function hpdf_write_parallel_data_interleaved_mode

Function name	hpdf_write_parallel_data_interleaved_mode
Function prototype	void hpdf_write_parallel_data_interleaved_mode(hpdf_channel_enum channelx, int32_t data);
Function descriptions	write the parallel data on interleaved mode of data packing
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
data	the parallel data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write the parallel data on interleaved mode of data packing */
```

```
hpdf_write_parallel_data_interleaved_mode(CHANNEL0, 0xEFFFFFFF);
```

hpdf_write_parallel_data_dual_mode

The description of hpdf_write_parallel_data_dual_mode is shown as below:

Table 3-374. Function hpdf_write_parallel_data_dual_mode

Function name	hpdf_write_parallel_data_dual_mode
Function prototype	void hpdf_write_parallel_data_dual_mode(hpdf_channel_enum channelx, int32_t data);
Function descriptions	write the parallel data on dual mode of data packing
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
data	the parallel data
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* write the parallel data on dual mode of data packing */
```

```
hpdf_write_parallel_data_dual_mode(CHANNEL0, 0xEFFFFFFF);
```

hpdf_pulse_skip_update

The description of hpdf_pulse_skip_update is shown as below:

Table 3-375. Function hpdf_pulse_skip_update

Function name	hpdf_pulse_skip_update
Function prototype	void hpdf_pulse_skip_update(hpdf_channel_enum channelx, uint8_t number);
Function descriptions	update the number of pulses to skip
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
number	the number of serial input samples that will be skipped
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* update the number of pulses to skip */
```

```
pdf_pulse_skip_update(CHANNEL0, 63);
```

hpdf_pulse_skip_read

The description of hpdf_pulse_skip_read is shown as below:

Table 3-376. Function hpdf_pulse_skip_read

Function name	hpdf_pulse_skip_read
Function prototype	uint8_t hpdf_pulse_skip_read(hpdf_channel_enum channelx);
Function descriptions	read the number of pulses to skip
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum

Output parameter{out}	
-	-
Return value	
uint8_t	the number of pulses to skip

Example:

```
/* read the number of pulses to skip */

uint8_t value;

value = hpdf_pulse_skip_read(CHANNEL0);
```

hpdf_filter_enable

The description of hpdf_filter_enable is shown as below:

Table 3-377. Function hpdf_filter_enable

Function name	hpdf_filter_enable
Function prototype	void hpdf_filter_enable(hpdf_filter_enum filtery);
Function descriptions	enable filter
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..1)	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable filter0 */

hpdf_filter_enable(FLT0);
```

hpdf_filter_disable

The description of hpdf_filter_disable is shown as below:

Table 3-378. Function hpdf_filter_disable

Function name	hpdf_filter_disable
Function prototype	void hpdf_filter_disable(hpdf_filter_enum filtery);
Function descriptions	disable filter
Precondition	-
The called functions	-
Input parameter{in}	

filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable filter0 */
```

```
hpdf_filter_disable(FLT0);
```

hpdf_filter_config

The description of hpdf_filter_config is shown as below:

Table 3-379. Function hpdf_filter_config

Function name	hpdf_filter_config
Function prototype	void hpdf_filter_config(hpdf_filter_enum filtery, uint32_t order, uint16_t oversample);
Function descriptions	configure sinc filter order and oversample
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
order	sinc filter order
<i>FLT_FASTSINC</i>	FastSinc filter type
<i>FLT_SINC1</i>	Sinc1 filter type
<i>FLT_SINC2</i>	Sinc2 filter type
<i>FLT_SINC3</i>	Sinc3 filter type
<i>FLT_SINC4</i>	Sinc4 filter type
<i>FLT_SINC5</i>	Sinc5 filter type
Input parameter{in}	
oversample	Sinc filter oversampling rate(1-1024)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure sinc filter order and oversample */
```

```
hpdf_filter_config(FLT0, FLT_SINC2, 64);
```

hpdf_integrator_oversample

The description of hpdf_integrator_oversample is shown as below:

Table 3-380. Function hpdf_integrator_oversample

Function name	hpdf_integrator_oversample
Function prototype	void hpdf_integrator_oversample(hpdf_filter_enum filtery, uint8_t oversample);
Function descriptions	configure integrator oversampling rate
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
oversample	integrator oversampling rate(1-256)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure integrator oversampling rate */
hpdf_integrator_oversample(FLT0, 256);
```

hpdf_threshold_monitor_filter_config

The description of hpdf_threshold_monitor_filter_config is shown as below:

Table 3-381. Function hpdf_threshold_monitor_filter_config

Function name	hpdf_threshold_monitor_filter_config
Function prototype	void hpdf_threshold_monitor_filter_config(hpdf_channel_enum channelx, uint32_t order, uint8_t oversample);
Function descriptions	configure threshold monitor filter order and oversample
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Input parameter{in}	
order	threshold monitor Sinc filter order
<i>TM_FASTSINC</i>	FastSinc filter type
<i>TM_SINC1</i>	Sinc1 filter type
<i>TM_SINC2</i>	Sinc2 filter type

<i>TM_SINC3</i>	Sinc3 filter type
Input parameter{in}	
oversample	Sinc filter oversampling rate(1-32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor filter order and oversample */
```

```
hpdf_threshold_monitor_filter_config (FLT0, TM_SINC3);
```

hpdf_threshold_monitor_filter_read_data

The description of hpdf_threshold_monitor_filter_read_data is shown as below:

Table 3-382. Function hpdf_threshold_monitor_filter_read_data

Function name	hpdf_threshold_monitor_filter_read_data
Function prototype	int16_t hpdf_threshold_monitor_filter_read_data(hpdf_channel_enum channelx);
Function descriptions	read the threshold monitor filter data
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
int16_t	the threshold monitor filter data

Example:

```
/* read the threshold monitor filter data */
```

```
int16_t data;
```

```
data = hpdf_threshold_monitor_filter_read_data(CHANNEL0);
```

hpdf_threshold_monitor_fast_mode_disable

The description of hpdf_threshold_monitor_fast_mode_disable is shown as below:

Table 3-383. Function hpdf_threshold_monitor_fast_mode_disable

Function name	hpdf_threshold_monitor_fast_mode_disable
Function prototype	void hpdf_threshold_monitor_fast_mode_disable(hpdf_filter_enum filtery);

Function descriptions	disable threshold monitor fast mode
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable threshold monitor fast mode */
hpdf_threshold_monitor_fast_mode_disable (FLT0);
```

hpdf_threshold_monitor_fast_mode_enable

The description of hpdf_threshold_monitor_fast_mode_enable is shown as below:

Table 3-384. Function hpdf_threshold_monitor_fast_mode_enable

Function name	hpdf_threshold_monitor_fast_mode_enable
Function prototype	void hpdf_threshold_monitor_fast_mode_enable(hpdf_filter_enum filtery);
Function descriptions	enable threshold monitor fast mode
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable threshold monitor fast mode */
hpdf_threshold_monitor_fast_mode_enable(FLT0);
```

hpdf_threshold_monitor_channel

The description of hpdf_threshold_monitor_channel is shown as below:

Table 3-385. Function hpdf_threshold_monitor_channel

Function name	hpdf_threshold_monitor_channel
Function prototype	void hpdf_threshold_monitor_channel(hpdf_filter_enum filtery, uint32_t

	channel);
Function descriptions	configure threshold monitor channel
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
channel	which channel use threshold monitorx(x=0,1)
<i>TMCHEN_DISABLE</i>	threshold monitorx is disabled on channel 0 and channel 1
<i>TMCHEN_CHANNEL0</i>	threshold monitor x is enabled on channel 0
<i>TMCHEN_CHANNEL1</i>	threshold monitor x is enabled on channel 1
<i>TMCHEN_CHANNEL0_1</i>	threshold monitor x is enabled on channel 0 and channel 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor channel */
```

```
hpdf_threshold_monitor_channel(FLT0, TMCHEN_CHANNEL0_1);
```

hpdf_threshold_monitor_high_threshold

The description of hpdf_threshold_monitor_high_threshold is shown as below:

Table 3-386. Function hpdf_threshold_monitor_high_threshold

Function name	hpdf_threshold_monitor_high_threshold
Function prototype	void hpdf_threshold_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);
Function descriptions	configure threshold monitor high threshold value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
value	high threshold value(-8388608~8388607)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor high threshold value */

hpdf_threshold_monitor_high_threshold(FLT0, 32767);
```

hpdf_threshold_monitor_low_threshold

The description of hpdf_threshold_monitor_low_threshold is shown as below:

Table 3-387. Function hpdf_threshold_monitor_low_threshold

Function name	hpdf_threshold_monitor_low_threshold
Function prototype	void hpdf_threshold_monitor_low_threshold(hpdf_filter_enum filtery, int32_t value);
Function descriptions	configure threshold monitor low threshold value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
value	low threshold value(-8388608~8388607)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor low threshold value */

hpdf_threshold_monitor_low_threshold(FLT0, -32768);
```

hpdf_high_threshold_break_signal

The description of hpdf_high_threshold_break_signal is shown as below:

Table 3-388. Function hpdf_high_threshold_break_signal

Function name	hpdf_high_threshold_break_signal
Function prototype	void hpdf_high_threshold_break_signal(hpdf_filter_enum filtery, uint32_t break_signal);
Function descriptions	configure threshold monitor high threshold event break signal
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum

Input parameter{in}	
break_signal	HPDF break signal
<i>NO_TM_HT_BREAK</i>	break signal is not distributed to an threshold monitor high threshold event
<i>TM_HT_BREAK0</i>	break signal 0 is distributed to an threshold monitor high threshold event
<i>TM_HT_BREAK1</i>	break signal 1 is distributed to an threshold monitor high threshold event
<i>TM_HT_BREAK0_1</i>	break signal 0 and 1 is distributed to an threshold monitor high threshold event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor high threshold event break signal */
hpdf_high_threshold_break_signal(FTL0, TM_HT_BREAK0_1);
```

hpdf_low_threshold_break_signal

The description of hpdf_low_threshold_break_signal is shown as below:

Table 3-389. Function hpdf_low_threshold_break_signal

Function name	hpdf_low_threshold_break_signal
Function prototype	void hpdf_low_threshold_break_signal(hpdf_filter_enum filtery, uint32_t break_signal);
Function descriptions	configure threshold monitor low threshold event break signal
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
break_signal	HPDF break signal
<i>NO_TM_LT_BREAK</i>	break signal is not distributed to an threshold monitor low threshold event
<i>TM_LT_BREAK0</i>	break signal 0 is distributed to an threshold monitor low threshold event
<i>TM_LT_BREAK1</i>	break signal 1 is distributed to an threshold monitor low threshold event
<i>TM_LT_BREAK0_1</i>	break signal 0 and 1 is distributed to an threshold monitor low threshold event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor low threshold event break signal */
```

```
hpdf_low_threshold_break_signal(FTL0, TM_LT_BREAK0_1);
```

hpdf_extremes_monitor_channel

The description of hpdf_extremes_monitor_channel is shown as below:

Table 3-390. Function hpdf_extremes_monitor_channel

Function name	hpdf_extremes_monitor_channel
Function prototype	void hpdf_extremes_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);
Function descriptions	configure extremes monitor channel
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
channel	which channel use extremes monitor y (y=0,1)
<i>EM_CHANNEL_DISABLE</i>	extremes monitor y does not accept data from channel 0 and channel 1
<i>EM_CHANNEL0</i>	extremes monitor y accepts data from channel 0
<i>EM_CHANNEL1</i>	extremes monitor y accepts data from channel 1
<i>EM_CHANNEL0_1</i>	extremes monitor y accepts data from channel 0 and channel 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure extremes monitor channel */
```

```
hpdf_extremes_monitor_channel(FTL0, EM_CHANNEL0_1);
```

hpdf_extremes_monitor_maximum_get

The description of hpdf_extremes_monitor_maximum_get is shown as below:

Table 3-391. Function hpdf_extremes_monitor_maximum_get

Function name	hpdf_extremes_monitor_maximum_get
Function prototype	int32_t hpdf_extremes_monitor_maximum_get(hpdf_filter_enum filtery);
Function descriptions	get the extremes monitor maximum value
Precondition	-
The called functions	-
Input parameter{in}	

filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
int32_t	the maximum value

Example:

```
int32_t vlaue;
```

```
vlaue = hpdf_extremes_monitor_maximum_get(FTL0);
```

hpdf_extremes_monitor_minimum_get

The description of hpdf_extremes_monitor_minimum_get is shown as below:

Table 3-392. Function hpdf_extremes_monitor_minimum_get

Function name	hpdf_extremes_monitor_minimum_get
Function prototype	int32_t hpdf_extremes_monitor_minimum_get(hpdf_filter_enum filtery);
Function descriptions	get the extremes monitor minimum value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
int32_t	the minimum value

Example:

```
/* get the extremes monitor minimum value */
```

```
int32_t vlaue;
```

```
vlaue = hpdf_extremes_monitor_minimum_get(FTL0,);
```

hpdf_rc_continuous_disable

The description of hpdf_rc_continuous_disable is shown as below:

Table 3-393. Function hpdf_rc_continuous_disable

Function name	hpdf_rc_continuous_disable
Function prototype	void hpdf_rc_continuous_disable(hpdf_filter_enum filtery);
Function descriptions	disable regular conversions continuous mode
Precondition	-

The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..1)	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable regular conversions continuous mode */
```

```
hpdf_rc_continuous_disable(FTL0);
```

hpdf_rc_continuous_enable

The description of hpdf_rc_continuous_enable is shown as below:

Table 3-394. Function hpdf_rc_continuous_enable

Function name	hpdf_rc_continuous_enable
Function prototype	void hpdf_rc_continuous_enable(hpdf_filter_enum filtery);
Function descriptions	enable regular conversions continuous mode
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..1)	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversions continuous mode */
```

```
hpdf_rc_continuous_enable(FTL0);
```

hpdf_rc_start_by_software

The description of hpdf_rc_start_by_software is shown as below:

Table 3-395. Function hpdf_rc_start_by_software

Function name	hpdf_rc_start_by_software
Function prototype	void hpdf_rc_start_by_software(hpdf_filter_enum filtery);
Function descriptions	start regular channel conversion by software
Precondition	-

The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..1)	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start regular channel conversion by software */
```

```
hpdf_rc_start_by_software(FTL0);
```

hpdf_rc_syn_disable

The description of hpdf_rc_syn_disable is shown as below:

Table 3-396. Function hpdf_rc_syn_disable

Function name	hpdf_rc_syn_disable
Function prototype	void hpdf_rc_syn_disable(hpdf_filter_enum filtery);
Function descriptions	disable regular conversion synchronously
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..1)	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable regular conversion synchronously */
```

```
hpdf_rc_syn_disable(FTL0);
```

hpdf_rc_syn_enable

The description of hpdf_rc_syn_enable is shown as below:

Table 3-397. Function hpdf_rc_syn_enable

Function name	hpdf_rc_syn_enable
Function prototype	void hpdf_rc_syn_enable(hpdf_filter_enum filtery);
Function descriptions	enable regular conversion synchronously
Precondition	disable FLTy(y=0..1)

The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..1)	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversion synchronously */
```

```
hpdf_rc_syn_enable(FTL0);
```

hpdf_rc_dma_disable

The description of hpdf_rc_dma_disable is shown as below:

Table 3-398. Function hpdf_rc_dma_disable

Function name	hpdf_rc_dma_disable
Function prototype	void hpdf_rc_dma_disable(hpdf_filter_enum filtery);
Function descriptions	disable regular conversion DMA channel
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..1)	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable regular conversion DMA channel */
```

```
hpdf_rc_dma_disable(FTL0);
```

hpdf_rc_dma_enable

The description of hpdf_rc_dma_enable is shown as below:

Table 3-399. Function hpdf_rc_dma_enable

Function name	hpdf_rc_dma_enable
Function prototype	void hpdf_rc_dma_enable(hpdf_filter_enum filtery);
Function descriptions	enable regular conversion DMA channel
Precondition	disable FLTy(y=0..1)

The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversion DMA channel */
```

```
hpdf_rc_dma_enable(FTL0);
```

hpdf_rc_channel_config

The description of hpdf_rc_channel_config is shown as below:

Table 3-400. Function hpdf_rc_channel_config

Function name	hpdf_rc_channel_config
Function prototype	void hpdf_rc_channel_config(hpdf_filter_enum filtery, hpdf_channel_enum channelx);
Function descriptions	configure regular conversion channel
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
channelx	The channel of HPDF module
<i>CHANNELx(x=0..1)</i>	select HPDF channel, refer to Table 3-331. Enum hpdf_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure regular conversion channel */
```

```
hpdf_rc_channel_config(FTL0, CHANNEL1);
```

hpdf_rc_fast_mode_disable

The description of hpdf_rc_fast_mode_disable is shown as below:

Table 3-401. Function `hpdf_rc_fast_mode_disable`

Function name	<code>hpdf_rc_fast_mode_disable</code>
Function prototype	<code>void hpdf_rc_fast_mode_disable(hpdf_filter_enum filtery);</code>
Function descriptions	disable regular conversion fast conversion mode
Precondition	disable <code>FLTy(y=0..1)</code>
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<code>FLTy(y=0..1)</code>	select HPDF filter, refer to Table 3-332. Enum <code>hpdf_filter_enum</code>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable regular conversion fast conversion mode */
hpdf_rc_fast_mode_disable(FTL0);
```

`hpdf_rc_fast_mode_enable`

The description of `hpdf_rc_fast_mode_enable` is shown as below:

Table 3-402. Function `hpdf_rc_fast_mode_enable`

Function name	<code>hpdf_rc_fast_mode_enable</code>
Function prototype	<code>void hpdf_rc_fast_mode_enable(hpdf_filter_enum filtery);</code>
Function descriptions	enable regular conversion fast conversion mode
Precondition	disable <code>FLTy(y=0..1)</code>
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<code>FLTy(y=0..1)</code>	select HPDF filter, refer to Table 3-332. Enum <code>hpdf_filter_enum</code>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversion fast conversion mode */
hpdf_rc_fast_mode_enable(FTL0);
```

`hpdf_rc_data_get`

The description of `hpdf_rc_data_get` is shown as below:

Table 3-403. Function `hpdf_rc_data_get`

Function name	<code>hpdf_rc_data_get</code>
Function prototype	<code>int32_t hpdf_rc_data_get(hpdf_filter_enum filtery);</code>
Function descriptions	get the regular conversion data
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum <code>hpdf_filter_enum</code>
Output parameter{out}	
-	-
Return value	
int32_t	regular conversions data

Example:

```
/* get the regular conversion data */
int32_t vlaue;
vlaue = hpdf_rc_data_get(FTL0);
```

`hpdf_rc_channel_get`

The description of `hpdf_rc_channel_get` is shown as below:

Table 3-404. Function `hpdf_rc_channel_get`

Function name	<code>hpdf_rc_channel_get</code>
Function prototype	<code>uint8_t hpdf_rc_channel_get(hpdf_filter_enum filtery);</code>
Function descriptions	get the channel of regular channel most recently converted
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum <code>hpdf_filter_enum</code>
Output parameter{out}	
-	-
Return value	
uint8_t	the channel

Example:

```
/* get the channel of regular channel most recently converted */
uint8_t channel;
channel = hpdf_rc_channel_get(FTL0);
```

hpdf_ic_start_by_software

The description of hpdf_ic_start_by_software is shown as below:

Table 3-405. Function hpdf_ic_start_by_software

Function name	hpdf_ic_start_by_software
Function prototype	void hpdf_ic_start_by_software(hpdf_filter_enum filtery);
Function descriptions	start inserted channel conversion by software
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start inserted channel conversion by software */
hpdf_ic_start_by_software(FTL0);
```

hpdf_ic_syn_disable

The description of hpdf_ic_syn_disable is shown as below:

Table 3-406. Function hpdf_ic_syn_disable

Function name	hpdf_ic_syn_disable
Function prototype	void hpdf_ic_syn_disable(hpdf_filter_enum filtery);
Function descriptions	disable inserted conversion synchronously
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable inserted conversion synchronously */
hpdf_ic_syn_disable(FTL0);
```

hpdf_ic_syn_enable

The description of hpdf_ic_syn_enable is shown as below:

Table 3-407. Function hpdf_ic_syn_enable

Function name	hpdf_ic_syn_enable
Function prototype	void hpdf_ic_syn_enable(hpdf_filter_enum filtery);
Function descriptions	enable inserted conversion synchronously
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable inserted conversion synchronously */
hpdf_ic_syn_enable(FTL0);
```

hpdf_ic_dma_disable

The description of hpdf_ic_dma_disable is shown as below:

Table 3-408. Function hpdf_ic_dma_disable

Function name	hpdf_ic_dma_disable
Function prototype	void hpdf_ic_dma_disable(hpdf_filter_enum filtery);
Function descriptions	disable inserted conversion DMA channel
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable inserted conversion DMA channel */
hpdf_ic_dma_disable(FTL0);
```

hpdf_ic_dma_enable

The description of hpdf_ic_dma_enable is shown as below:

Table 3-409. Function hpdf_ic_dma_enable

Function name	hpdf_ic_dma_enable
Function prototype	void hpdf_ic_dma_enable(hpdf_filter_enum filtery);
Function descriptions	enable inserted conversion DMA channel
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable inserted conversion DMA channel */
hpdf_ic_dma_enable(FTL0);
```

hpdf_ic_scan_mode_disable

The description of hpdf_ic_scan_mode_disable is shown as below:

Table 3-410. Function hpdf_ic_scan_mode_disable

Function name	hpdf_ic_scan_mode_disable
Function prototype	void hpdf_ic_scan_mode_disable(hpdf_filter_enum filtery);
Function descriptions	disable scan conversion mode
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable scan conversion mode */
hpdf_ic_scan_mode_disable(FTL0);
```

hpdf_ic_scan_mode_enable

The description of hpdf_ic_scan_mode_enable is shown as below:

Table 3-411. Function hpdf_ic_scan_mode_enable

Function name	hpdf_ic_scan_mode_enable
Function prototype	void hpdf_ic_scan_mode_enable(hpdf_filter_enum filtery);
Function descriptions	enable scan conversion mode
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable scan conversion mode */
hpdf_ic_scan_mode_enable(FTL0);
```

hpdf_ic_trigger_signal_disbale

The description of hpdf_ic_trigger_signal_disbale is shown as below:

Table 3-412. Function hpdf_ic_trigger_signal_disbale

Function name	hpdf_ic_trigger_signal_disbale
Function prototype	void hpdf_ic_trigger_signal_disbale(hpdf_filter_enum filtery);
Function descriptions	disable inserted conversions trigger siganl
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable inserted conversions trigger siganl */
hpdf_ic_trigger_signal_disbale(FTL0);
```

hpdf_ic_trigger_signal_config

The description of hpdf_ic_trigger_signal_config is shown as below:

Table 3-413. Function hpdf_ic_trigger_signal_config

Function name	hpdf_ic_trigger_signal_config
Function prototype	void hpdf_ic_trigger_signal_config(hpdf_filter_enum filtery, uint32_t trigger, uint32_t trigger_edge);
Function descriptions	configure inserted conversions trigger signal and trigger edge
Precondition	disable FLTy(y=0..1)
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
trigger	inserted conversions trigger signal
<i>HPDF_ITRG0</i>	TIMER1_TRGO is selected to start inserted conversion
<i>HPDF_ITRG1</i>	TIMER2_TRGO is selected to start inserted conversion
<i>HPDF_ITRG2</i>	TIMER3_TRGO is selected to start inserted conversion
<i>HPDF_ITRG3</i>	TIMER4_TRGO is selected to start inserted conversion
<i>HPDF_ITRG24</i>	EXTI11 is selected to start inserted conversion
<i>HPDF_ITRG25</i>	EXTI15 is selected to start inserted conversion
<i>HPDF_ITRG26</i>	TIMER5_TRGO is selected to start inserted conversion
Input parameter{in}	
trigger_edge	inserted conversions trigger edge
<i>TRG_DISABLE</i>	disable trigger signal
<i>RISING_EDGE_TRG</i>	rising edge on the trigger signal
<i>FALLING_EDGE_TRG</i>	falling edge on the trigger signal
<i>EDGE_TRG</i>	edge (rising edges and falling edges) on the trigger signal
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure inserted conversions trigger signal and trigger edge */
hpdf_ic_trigger_signal_config(FTL0, HPDF_ITRG3, FALLING_EDGE_TRG);
```

hpdf_ic_channel_config

The description of hpdf_ic_channel_config is shown as below:

Table 3-414. Function hpdf_ic_channel_config

Function name	hpdf_ic_channel_config
----------------------	------------------------

Function prototype	void hpdf_ic_channel_config(hpdf_filter_enum filtery, uint32_t channel);
Function descriptions	configure inserted group conversions channel
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
channel	the HPDF channel belongs to inserted group
<i>IGCSEL_CHANNEL0</i>	channel0 belongs to the inserted group
<i>IGCSEL_CHANNEL1</i>	channel1 belongs to the inserted group
<i>IGCSEL_CHANNEL0_1</i>	channel0 and channel1 belongs to the inserted group
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure inserted group conversions channel */
hpdf_ic_channel_config(FTL0, IGCSEL_CHANNEL0_1);
```

hpdf_ic_data_get

The description of hpdf_ic_data_get is shown as below:

Table 3-415. Function hpdf_ic_data_get

Function name	hpdf_ic_data_get
Function prototype	int32_t hpdf_ic_data_get(hpdf_filter_enum filtery);
Function descriptions	get the inserted conversions data
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
int32_t	inserted conversions data

Example:

```
/* get the inserted conversions data */
int32_t vlaue;
```

```
vlaue = hpdf_ic_data_get(FTL0);
```

hpdf_ic_channel_get

The description of hpdf_ic_channel_get is shown as below:

Table 3-416. Function hpdf_ic_channel_get

Function name	hpdf_ic_channel_get
Function prototype	uint8_t hpdf_ic_channel_get(hpdf_filter_enum filtery);
Function descriptions	get the channel of inserted group channel most recently converted
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Output parameter{out}	
-	-
Return value	
uint8_t	the channel

Example:

```
/* get the channel of inserted group channel most recently converted */
```

```
uint8_t channel;
```

```
channel = hpdf_ic_channel_get(FTL0);
```

hpdf_flag_get

The description of hpdf_flag_get is shown as below:

Table 3-417. Function hpdf_flag_get

Function name	hpdf_flag_get
Function prototype	FlagStatus hpdf_flag_get(hpdf_filter_enum filtery, hpdf_flag_enum flag);
Function descriptions	get the HPDF flags
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
flag	HPDF flags, refer to Table 3-333. Enum hpdf_flag_enum
<i>HPDF_FLAG_FLTy_IC EF</i>	FLTy inserted conversion end flag
<i>HPDF_FLAG_FLTy_RC EF</i>	FLTy regular conversion end flag

<i>HPDF_FLAG_FLTy_IC DOF</i>	FLTy inserted conversion data overflow flag
<i>HPDF_FLAG_FLTy_RC DOF</i>	FLTy threshold monitor event occurred flag
<i>HPDF_FLAG_FLTy_TM EOF</i>	FLTy inserted conversion in progress flag
<i>HPDF_FLAG_FLTy_IC PF</i>	FLTy regular conversion in progress flag
<i>HPDF_FLAG_FLTy_RC PF</i>	clock signal is lost on channel 0 flag
<i>HPDF_FLAG_FLT0_CK LF0</i>	clock signal is lost on channel 1 flag
<i>HPDF_FLAG_FLT0_CK LF1</i>	malfunction event occurred on channel 0 flag
<i>HPDF_FLAG_FLT0_M MF0</i>	malfunction occurred on channel 1 flag
<i>HPDF_FLAG_FLT0_M MF1</i>	FLTy inserted channel most recently converted
<i>HPDF_FLAG_FLTy_RC HPDT</i>	FLTy inserted channel most recently converted
<i>HPDF_FLAG_FLTy_LT F0</i>	threshold monitor low threshold flag on channel 0 flag
<i>HPDF_FLAG_FLTy_LT F1</i>	threshold monitor low threshold flag on channel 1 flag
<i>HPDF_FLAG_FLTy_HT F0</i>	threshold monitor high threshold flag on channel 0 flag
<i>HPDF_FLAG_FLTy_HT F1</i>	threshold monitor high threshold flag on channel 1 flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the FLT0 threshold monitor event occurred flag */
```

```
hpdf_flag_get(FTL0, HPDF_FLAG_FLTy_TMEOF);
```

hpdf_flag_clear

The description of hpdf_flag_clear is shown as below:

Table 3-418. Function hpdf_flag_clear

Function name	hpdf_flag_clear
----------------------	-----------------

Function prototype	void hpdf_flag_clear(hpdf_filter_enum filtery, hpdf_flag_enum flag);
Function descriptions	clear the HPDF flags
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
flag	HPDF flags, refer to Table 3-333. Enum hpdf_flag_enum
<i>HPDF_FLAG_FLTy_IC EF</i>	FLTy inserted conversion end flag
<i>HPDF_FLAG_FLTy_RC EF</i>	FLTy regular conversion end flag
<i>HPDF_FLAG_FLTy_IC DOF</i>	FLTy inserted conversion data overflow flag
<i>HPDF_FLAG_FLTy_RC DOF</i>	FLTy threshold monitor event occurred flag
<i>HPDF_FLAG_FLTy_TM EOF</i>	FLTy inserted conversion in progress flag
<i>HPDF_FLAG_FLTy_IC PF</i>	FLTy regular conversion in progress flag
<i>HPDF_FLAG_FLTy_RC PF</i>	clock signal is lost on channel 0 flag
<i>HPDF_FLAG_FLT0_CK LF0</i>	clock signal is lost on channel 1 flag
<i>HPDF_FLAG_FLT0_CK LF1</i>	malfunction event occurred on channel 0 flag
<i>HPDF_FLAG_FLT0_M MF0</i>	malfunction occurred on channel 1 flag
<i>HPDF_FLAG_FLT0_M MF1</i>	FLTy inserted channel most recently converted
<i>HPDF_FLAG_FLTy_RC HPDT</i>	FLTy inserted channel most recently converted
<i>HPDF_FLAG_FLTy_LT F0</i>	threshold monitor low threshold flag on channel 0 flag
<i>HPDF_FLAG_FLTy_LT F1</i>	threshold monitor low threshold flag on channel 1 flag
<i>HPDF_FLAG_FLTy_HT F0</i>	threshold monitor high threshold flag on channel 0 flag
<i>HPDF_FLAG_FLTy_HT F1</i>	threshold monitor high threshold flag on channel 1 flag
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* clear the FLT0 threshold monitor event occurred flag */
```

```
hpdf_flag_clear(FLT0, HPDF_FLAG_FLTy_TMEOF);
```

hpdf_interrupt_enable

The description of hpdf_interrupt_enable is shown as below:

Table 3-419. Function hpdf_interrupt_enable

Function name	hpdf_interrupt_enable
Function prototype	void hpdf_interrupt_enable(hpdf_filter_enum filtery, hpdf_interrput_enum interrupt);
Function descriptions	enable HPDF interrupt
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
interrupt	HPDF interrupts, refer to Table 3-335. Enum hpdf_interrput_enum
HPDF_INT_FLTy_ICEI E	FLTy inserted conversion end interrupt enable
HPDF_INT_FLTy_RCEI E	FLTy regular conversion end interrupt enable
HPDF_INT_FLTy_ICD OIE	FLTy inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_RCD OIE	FLTy regular conversion data overflow interrupt enable
HPDF_INT_FLTy_TMIE	FLTy threshold monitor interrupt enable
HPDF_INT_FLT0_MMI E	malfunction monitor interrupt enable
HPDF_INT_FLT0_CKLI E	clock loss interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable FLT0 threshold monitor interrupt */
```

hpdf_interrupt_enable(FTL0, HPDF_INT_FLTy_TME0IE);

hpdf_interrupt_disable

The description of hpdf_interrupt_disable is shown as below:

Table 3-420. Function hpdf_interrupt_disable

Function name	hpdf_interrupt_disable
Function prototype	void hpdf_interrupt_disable(hpdf_filter_enum filtery, hpdf_interrput_enum interrupt);
Function descriptions	disable HPDF interrupt
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<i>FLTy(y=0..1)</i>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
interrupt	HPDF interrupts, refer to Table 3-335. Enum hpdf_interrput_enum
HPDF_INT_FLTy_ICEI E	FLTy inserted conversion end interrupt enable
HPDF_INT_FLTy_RCEI E	FLTy regular conversion end interrupt enable
HPDF_INT_FLTy_ICD OIE	FLTy inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_RCD OIE	FLTy regular conversion data overflow interrupt enable
HPDF_INT_FLTy_TMIE	FLTy threshold monitor interrupt enable
HPDF_INT_FLT0_MMI E	malfunction monitor interrupt enable
HPDF_INT_FLT0_CKLI E	clock loss interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable FLT0 threshold monitor interrupt */
```

```
hpdf_interrupt_disable(FTL0, HPDF_INT_FLT0_CKLIIE);
```

hpdf_interrupt_flag_get

The description of hpdf_interrupt_flag_get is shown as below:

Table 3-421. Function `hpdf_interrupt_flag_get`

Function name	<code>hpdf_interrupt_flag_get</code>
Function prototype	<code>FlagStatus hpdf_interrupt_flag_get(hpdf_filter_enum filtery, hpdf_interrupt_flag_enum int_flag);</code>
Function descriptions	get the HPDF interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<code>FLTy(y=0..1)</code>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
int_flag	HPDF flags, refer to Table 3-334. Enum hpdf_interrput_flag_enum
<code>HPDF_INT_FLAG_FLTy_ICEF</code>	FLTy inserted conversion end interrupt flag
<code>HPDF_INT_FLAG_FLTy_RCEF</code>	FLTy regular conversion end interrupt flag
<code>HPDF_INT_FLAG_FLTy_ICDOF</code>	FLTy inserted conversion data overflow interrupt flag
<code>HPDF_INT_FLAG_FLTy_RCDOF</code>	FLTy regular conversion data overflow interrupt flag
<code>HPDF_INT_FLAG_FLTy_TMEOF</code>	FLTy threshold monitor event occurred interrupt flag
<code>HPDF_INT_FLAG_FLT0_CKLF0</code>	clock signal is lost on channel 0 interrupt flag
<code>HPDF_INT_FLAG_FLT0_CKLF1</code>	clock signal is lost on channel 1 interrupt flag
<code>HPDF_INT_FLAG_FLT0_MMF0</code>	malfunction event occurred on channel 0 interrupt flag
<code>HPDF_INT_FLAG_FLT0_MMF1</code>	malfunction event occurred on channel 1 interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the the clock loss interrupt flag */
```

```
hpdf_interrupt_flag_get (FTL0, HPDF_INT_FLAG_FLT0_CKLF0);
```

hpdf_interrupt_flag_clear

The description of `hpdf_interrupt_flag_clear` is shown as below:

Table 3-422. Function `hpdf_interrupt_flag_clear`

Function name	<code>hpdf_interrupt_flag_clear</code>
Function prototype	<code>void hpdf_interrupt_flag_clear(hpdf_filter_enum filtery, hpdf_interrupt_flag_enum int_flag);</code>
Function descriptions	clear the HPDF interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
<code>FLTy(y=0..1)</code>	select HPDF filter, refer to Table 3-332. Enum hpdf_filter_enum
Input parameter{in}	
int_flag	HPDF flags, refer to Table 3-334. Enum hpdf_interrput_flag_enum
<code>HPDF_INT_FLAG_FLT_y_ICEF</code>	FLTy inserted conversion end interrupt flag
<code>HPDF_INT_FLAG_FLT_y_RCEF</code>	FLTy regular conversion end interrupt flag
<code>HPDF_INT_FLAG_FLT_y_ICDOF</code>	FLTy inserted conversion data overflow interrupt flag
<code>HPDF_INT_FLAG_FLT_y_RCDOF</code>	FLTy regular conversion data overflow interrupt flag
<code>HPDF_INT_FLAG_FLT_y_TMEOF</code>	FLTy threshold monitor event occurred interrupt flag
<code>HPDF_INT_FLAG_FLT_0_CKLF0</code>	clock signal is lost on channel 0 interrupt flag
<code>HPDF_INT_FLAG_FLT_0_CKLF1</code>	clock signal is lost on channel 1 interrupt flag
<code>HPDF_INT_FLAG_FLT_0_MMF0</code>	malfunction event occurred on channel 0 interrupt flag
<code>HPDF_INT_FLAG_FLT_0_MMF1</code>	malfunction event occurred on channel 1 interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the the clock loss interrupt flag */
```

```
hpdf_interrupt_flag_get(FTL0, HPDF_INT_FLAG_FLT0_CKLF0);
```

3.15. 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 [3.15.1](#), the I2C firmware functions are introduced in chapter [3.15.2](#).

3.15.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

Table 3-423. 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_TIMING	Timing register
I2C_TIMEOUT	Timeout register
I2C_STAT	Status register
I2C_STATC	I2C status clear register
I2C_PEC	PEC register
I2C_RDATA	Receive data register
I2C_TDATA	Transmit data register
I2C_CTL2	Control register 2

3.15.2. Descriptions of Peripheral functions

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

Table 3-424. I2C firmware function

Function name	Function description
i2c_deinit	reset I2C
i2c_timing_config	configure the timing parameters
i2c_digital_noise_filter_config	configure digital noise filter
i2c_analog_noise_filter_enable	enable analog noise filter
i2c_analog_noise_filter_disable	disable analog noise filter
i2c_master_clock_config	configure the SCL high and low period of clock in master mode
i2c_master_addressing	configure i2c slave addresss and transfer direction in master mode
i2c_address10_header_enable	10-bit address header executes read direction only in master receive mode
i2c_address10_header_disable	10-bit address header executes complete sequence in master receive mode
i2c_address10_enable	enable 10-bit addressing mode in master mode
i2c_address10_disable	disable 10-bit addressing mode in master mode

Function name	Function description
i2c_automatic_end_enable	enable I2C automatic end mode in master mode
i2c_automatic_end_disable	disable I2C automatic end mode in master mode
i2c_slave_response_to_gcall_enable	enable the response to a general call
i2c_slave_response_to_gcall_disable	disable the response to a general call
i2c_stretch_scl_low_enable	enable to stretch SCL low when data is not ready in slave mode
i2c_stretch_scl_low_disable	disable to stretch SCL low when data is not ready in slave mode
i2c_address_config	configure i2c slave address
i2c_address_bit_compare_config	define which bits of ADDRESS[7:1] need to compare with the incoming address byte
i2c_address_disable	disable i2c address in slave mode
i2c_second_address_config	configure i2c second slave address
i2c_second_address_disable	disable i2c second address in slave mode
i2c_received_address_get	get received match address in slave mode
i2c_slave_byte_control_enable	enable slave byte control
i2c_slave_byte_control_disable	disable slave byte control
i2c_nack_enable	generate a NACK in slave mode
i2c_nack_disable	generate an ACK in slave mode
i2c_wakeup_from_deepsleep_enable	enable wakeup from Deep-sleep mode
i2c_wakeup_from_deepsleep_disable	disable wakeup from Deep-sleep 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
i2c_data_receive	I2C receive data
i2c_reload_enable	enable I2C reload mode
i2c_reload_disable	disable I2C reload mode
i2c_transfer_byte_number_config	configure number of bytes to be transferred
i2c_dma_enable	enable I2C DMA for transmission or reception
i2c_dma_disable	disable I2C DMA for transmission or reception
i2c_pec_transfer	I2C transfers PEC value
i2c_pec_enable	enable I2C PEC calculation
i2c_pec_disable	disable I2C PEC calculation
i2c_pec_value_get	get packet error checking value
i2c_smbus_alert_enable	enable SMBus Alert
i2c_smbus_alert_disable	disable SMBus Alert
i2c_smbus_default_addr_enable	enable SMBus device default address
i2c_smbus_default_addr_disable	disable SMBus device default address
i2c_smbus_host_addr_enable	enable SMBus Host address

Function name	Function description
i2c_smbus_host_addr_disable	disable SMBus Host address
i2c_extented_clock_timeout_enable	enable extended clock timeout detection
i2c_extented_clock_timeout_disable	disable extended clock timeout detection
i2c_clock_timeout_enable	enable clock timeout detection
i2c_clock_timeout_disable	disable clock timeout detection
i2c_bus_timeout_b_config	configure bus timeout B
i2c_bus_timeout_a_config	configure bus timeout A
i2c_idle_clock_timeout_config	configure idle clock timeout detection
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_interrupt_flag_enum

Table 3-425. i2c_interrupt_flag_enum

Member name	Function description
I2C_INT_FLAG_TI	transmit interrupt flag
I2C_INT_FLAG_RBNE	I2C_RDATA is not empty during receiving interrupt flag
I2C_INT_FLAG_ADDSEND	address received matches in slave mode interrupt flag
I2C_INT_FLAG_NACK	not acknowledge interrupt flag
I2C_INT_FLAG_STPDET	stop condition detected in slave mode interrupt flag
I2C_INT_FLAG_TC	transfer complete in master mode interrupt flag
I2C_INT_FLAG_TCR	transfer complete reload interrupt flag
I2C_INT_FLAG_BERR	bus error interrupt flag
I2C_INT_FLAG_LOSTARB	arbitration lost interrupt flag
I2C_INT_FLAG_OUERR	overflow/underrun error in slave mode interrupt flag
I2C_INT_FLAG_PECERR	PEC error interrupt flag
I2C_INT_FLAG_TIMEOUT	timeout interrupt flag
I2C_INT_FLAG_SMBALT	SMBus Alert interrupt flag

i2c_deinit

The description of i2c_deinit is shown as below:

Table 3-426. 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
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset I2C0 */
```

```
i2c_deinit(I2C0);
```

i2c_timing_config

The description of i2c_timing_config is shown as below:

Table 3-427. Function i2c_timing_config

Function name	i2c_timing_config
Function prototype	void i2c_timing_config(uint32_t i2c_periph, uint32_t psc, uint32_t scl_dely, uint32_t sda_dely);
Function descriptions	configure the timing parameters
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
psc	0-0xf, timing prescaler
Input parameter{in}	
scl_dely	0-0xf, data setup time
Input parameter{in}	
sda_dely	0-0xf, data hold time
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the timing parameters */
```

```
i2c_timing_config(I2C0, 0x1, 0x2, 0x1);
```

i2c_digital_noise_filter_config

The description of i2c_digital_noise_filter_config is shown as below:

Table 3-428. Function i2c_digital_noise_filter_config

Function name	i2c_digital_noise_filter_config
Function prototype	void i2c_digital_noise_filter_config(uint32_t i2c_periph, uint32_t filter_length);
Function descriptions	configure digital noise filter
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
filter_length	filter_length
<i>FILTER_DISABLE</i>	digital filter is disabled
<i>FILTER_LENGTH_1</i>	digital filter is enabled and filter spikes with a length of up to 1 tI2CCLK
<i>FILTER_LENGTH_2</i>	digital filter is enabled and filter spikes with a length of up to 2 tI2CCLK
<i>FILTER_LENGTH_3</i>	digital filter is enabled and filter spikes with a length of up to 3 tI2CCLK
<i>FILTER_LENGTH_4</i>	digital filter is enabled and filter spikes with a length of up to 4 tI2CCLK
<i>FILTER_LENGTH_5</i>	digital filter is enabled and filter spikes with a length of up to 5 tI2CCLK
<i>FILTER_LENGTH_6</i>	digital filter is enabled and filter spikes with a length of up to 6 tI2CCLK
<i>FILTER_LENGTH_7</i>	digital filter is enabled and filter spikes with a length of up to 7 tI2CCLK
<i>FILTER_LENGTH_8</i>	digital filter is enabled and filter spikes with a length of up to 8 tI2CCLK
<i>FILTER_LENGTH_9</i>	digital filter is enabled and filter spikes with a length of up to 9 tI2CCLK
<i>FILTER_LENGTH_10</i>	digital filter is enabled and filter spikes with a length of up to 10 tI2CCLK
<i>FILTER_LENGTH_11</i>	digital filter is enabled and filter spikes with a length of up to 11 tI2CCLK
<i>FILTER_LENGTH_12</i>	digital filter is enabled and filter spikes with a length of up to 12 tI2CCLK
<i>FILTER_LENGTH_13</i>	digital filter is enabled and filter spikes with a length of up to 13 tI2CCLK
<i>FILTER_LENGTH_14</i>	digital filter is enabled and filter spikes with a length of up to 14 tI2CCLK
<i>FILTER_LENGTH_15</i>	digital filter is enabled and filter spikes with a length of up to 15 tI2CCLK
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 digital filter filters spikes with a length of up to 1 tI2CCLK */
```

```
i2c_digital_noise_filter_config(I2C0, FILTER_LENGTH_1);
```

i2c_analog_noise_filter_enable

The description of i2c_analog_noise_filter_enable is shown as below:

Table 3-429. Function i2c_analog_noise_filter_enable

Function name	i2c_analog_noise_filter_enable
Function prototype	void i2c_analog_noise_filter_enable(uint32_t i2c_periph);
Function descriptions	enable analog noise filter
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable analog noise filter */
i2c_analog_noise_filter_enable(I2C0);
```

i2c_analog_noise_filter_disable

The description of i2c_analog_noise_filter_disable is shown as below:

Table 3-430. Function i2c_analog_noise_filter_disable

Function name	i2c_analog_noise_filter_disable
Function prototype	void i2c_analog_noise_filter_disable(uint32_t i2c_periph);
Function descriptions	disable analog noise filter
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable analog noise filter */
i2c_analog_noise_filter_disable(I2C0);
```

i2c_master_clock_config

The description of i2c_master_clock_config is shown as below:

Table 3-431. Function i2c_master_clock_config

Function name	i2c_master_clock_config
Function prototype	void i2c_master_clock_config(uint32_t i2c_periph, uint32_t sclh, uint32_t scll);
Function descriptions	configure the SCL high and low period of clock in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
sclh	0-0xff, SCL high period
Input parameter{in}	
scll	0-0xff, SCL low period
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the SCL and SDA period of clock in master mode */
```

```
i2c_master_clock_config (I2C0, 0x0f, 0x0f);
```

i2c_master_addressing

The description of i2c_master_addressing is shown as below:

Table 3-432. Function i2c_master_addressing

Function name	i2c_master_addressing
Function prototype	void i2c_master_addressing(uint32_t i2c_periph, uint32_t address, uint32_t trans_direction);
Function descriptions	configure i2c slave addresss and transfer direction in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
address	0-0x3FF except reserved address, I2C slave address to be sent
Input parameter{in}	

trans_direction	I2C transfer direction in master mode
<i>I2C_MASTER_TRANSMIT</i>	master transmit
<i>I2C_MASTER_RECEIVE</i>	master receive
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* send slave address to I2C bus */
```

```
i2c_master_addressing (I2C0, 0x82, I2C_MASTER_TRANSMIT);
```

i2c_address10_header_enable

The description of i2c_address10_header_enable is shown as below:

Table 3-433. Function i2c_address10_header_enable

Function name	i2c_address10_header_enable
Function prototype	void i2c_address10_header_enable(uint32_t i2c_periph);
Function descriptions	10-bit address header executes read direction only in master receive mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* 10-bit address header executes read direction only in master receive mode */
```

```
i2c_address10_header_enable(I2C0);
```

i2c_address10_header_disable

The description of i2c_address10_header_disable is shown as below:

Table 3-434. Function i2c_address10_header_disable

Function name	i2c_address10_header_disable
Function prototype	void i2c_address10_header_disable(uint32_t i2c_periph);
Function descriptions	10-bit address header executes complete sequence in master receive mode

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

Example:

```
/* 10-bit address header executes complete sequence in master receive mode */
```

```
i2c_address10_header_disable(I2C0);
```

i2c_address10_enable

The description of i2c_address10_enable is shown as below:

Table 3-435. Function i2c_address10_enable

Function name	i2c_address10_enable
Function prototype	void i2c_address10_enable(uint32_t i2c_periph);
Function descriptions	enable 10-bit addressing mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable 10-bit addressing mode in master mode */
```

```
i2c_address10_enable(I2C0);
```

i2c_address10_disable

The description of i2c_address10_disable is shown as below:

Table 3-436. Function i2c_address10_disable

Function name	i2c_address10_disable
Function prototype	void i2c_address10_disable(uint32_t i2c_periph);
Function descriptions	disable 10-bit addressing mode in master mode

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

Example:

```
/* disable 10-bit addressing mode in master mode */
```

```
i2c_address10_disable(I2C0);
```

i2c_automatic_end_enable

The description of i2c_automatic_end_enable is shown as below:

Table 3-437. Function i2c_automatic_end_enable

Function name	i2c_automatic_end_enable
Function prototype	void i2c_automatic_end_enable(uint32_t i2c_periph);
Function descriptions	enable I2C automatic end mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C automatic end mode in master mode */
```

```
i2c_automatic_end_enable(I2C0);
```

i2c_automatic_end_disable

The description of i2c_automatic_end_disable is shown as below:

Table 3-438. Function i2c_automatic_end_disable

Function name	i2c_automatic_end_disable
Function prototype	void i2c_automatic_end_disable(uint32_t i2c_periph);
Function descriptions	disable I2C automatic end mode in master mode

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

Example:

```
/* disable I2C automatic end mode in master mode */
```

```
i2c_automatic_end_disable(I2C0);
```

i2c_slave_response_to_gcall_enable

The description of i2c_slave_response_to_gcall_enable is shown as below:

Table 3-439. Function i2c_slave_response_to_gcall_enable

Function name	i2c_slave_response_to_gcall_enable
Function prototype	void i2c_slave_response_to_gcall_enable(uint32_t i2c_periph);
Function descriptions	enable the response to a general call
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the response to a general call */
```

```
i2c_slave_response_to_gcall_enable(I2C0);
```

i2c_slave_response_to_gcall_disable

The description of i2c_slave_response_to_gcall_disable is shown as below:

Table 3-440. Function i2c_slave_response_to_gcall_disable

Function name	i2c_slave_response_to_gcall_disable
Function prototype	void i2c_slave_response_to_gcall_disable(uint32_t i2c_periph);
Function descriptions	disable the response to a general call

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

Example:

```
/* disable the response to a general call */
```

```
i2c_slave_response_to_gcall_disable(I2C0);
```

i2c_stretch_scl_low_enable

The description of i2c_stretch_scl_low_enable is shown as below:

Table 3-441. Function i2c_stretch_scl_low_enable

Function name	i2c_stretch_scl_low_enable
Function prototype	void i2c_stretch_scl_low_enable(uint32_t i2c_periph);
Function descriptions	enable to stretch SCL low when data is not ready in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
i2c_stretch_scl_low_enable(I2C0);
```

i2c_stretch_scl_low_disable

The description of i2c_stretch_scl_low_disable is shown as below:

Table 3-442. Function i2c_stretch_scl_low_disable

Function name	i2c_stretch_scl_low_disable
Function prototype	void i2c_stretch_scl_low_disable(uint32_t i2c_periph);
Function descriptions	disable to stretch SCL low when data is not ready in slave mode

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

Example:

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

```
i2c_stretch_scl_low_disable(I2C0);
```

i2c_address_config

The description of i2c_address_config is shown as below:

Table 3-443. Function i2c_address_config

Function name	i2c_address_config
Function prototype	void i2c_address_config(uint32_t i2c_periph, uint32_t address, uint32_t addr_format);
Function descriptions	configure i2c slave address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
address	I2C address
Input parameter{in}	
addr_format	7bits or 10bits
<i>I2C_ADDFORMAT_7BITS</i>	7bits
<i>I2C_ADDFORMAT_10BITS</i>	10bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure i2c slave address */
```

i2c_address_config (I2C0, 0x82, I2C_ADDFORMAT_7BITS);

i2c_address_bit_compare_config

The description of i2c_address_bit_compare_config is shown as below:

Table 3-444. Function i2c_address_bit_compare_config

Function name	i2c_address_bit_compare_config
Function prototype	void i2c_address_bit_compare_config(uint32_t i2c_periph, uint32_t compare_bits);
Function descriptions	define which bits of ADDRESS[7:1] need to compare with the incoming address byte
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
compare_bits	the bits need to compare
<i>ADDRESS_BIT1_COMPARE</i>	address bit1 needs compare
<i>ADDRESS_BIT2_COMPARE</i>	address bit2 needs compare
<i>ADDRESS_BIT3_COMPARE</i>	address bit3 needs compare
<i>ADDRESS_BIT4_COMPARE</i>	address bit4 needs compare
<i>ADDRESS_BIT5_COMPARE</i>	address bit5 needs compare
<i>ADDRESS_BIT6_COMPARE</i>	address bit6 needs compare
<i>ADDRESS_BIT7_COMPARE</i>	address bit7 needs compare
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* bit 1 of ADDRESS[7:1] need to compare with the incoming address byte */

i2c_address_bit_compare_config(I2C0, ADDRESS_BIT1_COMPARE);

i2c_address_disable

The description of i2c_address_disable is shown as below:

Table 3-445. Function i2c_address_disable

Function name	i2c_address_disable
Function prototype	void i2c_address_disable(uint32_t i2c_periph);
Function descriptions	disable i2c address in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable i2c address in slave mode */
```

```
i2c_address_disable(I2C0);
```

i2c_second_address_config

The description of i2c_second_address_config is shown as below:

Table 3-446. Function i2c_second_address_config

Function name	i2c_second_address_config
Function prototype	void i2c_second_address_config(uint32_t i2c_periph, uint32_t address, uint32_t addr_mask);
Function descriptions	configure i2c second slave address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
address	I2C address
Input parameter{in}	
addr_mask	the bits not need to compare
<i>ADDRESS2_NO_MASK</i>	no mask, all the bits must be compared
<i>ADDRESS2_MASK_BIT1</i>	ADDRESS2[1] is masked, only ADDRESS2[7:2] are compared

ADDRESS2_MASK_BIT1_2	ADDRESS2[2:1] is masked, only ADDRESS2[7:3] are compared
ADDRESS2_MASK_BIT1_3	ADDRESS2[3:1] is masked, only ADDRESS2[7:4] are compared
ADDRESS2_MASK_BIT1_4	ADDRESS2[4:1] is masked, only ADDRESS2[7:5] are compared
ADDRESS2_MASK_BIT1_5	ADDRESS2[5:1] is masked, only ADDRESS2[7:6] are compared
ADDRESS2_MASK_BIT1_6	ADDRESS2[6:1] is masked, only ADDRESS2[7] are compared
ADDRESS2_MASK_ALL	all the ADDRESS2[7:1] bits are masked
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure i2c second slave address */
```

```
i2c_second_address_config(I2C0, 0x82, ADDRESS2_MASK_BIT1_2);
```

i2c_second_address_disable

The description of i2c_second_address_disable is shown as below:

Table 3-447. Function i2c_second_address_disable

Function name	i2c_second_address_disable
Function prototype	void i2c_second_address_disable(uint32_t i2c_periph);
Function descriptions	disable i2c second address in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable i2c second address in slave mode */
```

```
i2c_second_address_disable(I2C0);
```


i2c_receved_address_get

The description of i2c_receved_address_get is shown as below:

Table 3-448. Function i2c_receved_address_get

Function name	i2c_receved_address_get
Function prototype	uint32_t i2c_receved_address_get(uint32_t i2c_periph);
Function descriptions	get received match address in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
uint32_t	0x00..0x7F

Example:

```
/* get received match address in slave mode */
```

```
uint32_t address;
```

```
address = i2c_receved_address_get(I2C0);
```

i2c_slave_byte_control_enable

The description of i2c_slave_byte_control_enable is shown as below:

Table 3-449. Function i2c_slave_byte_control_enable

Function name	i2c_slave_byte_control_enable
Function prototype	void i2c_slave_byte_control_enable(uint32_t i2c_periph);
Function descriptions	enable slave byte control
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable slave byte control */
```

i2c_slave_byte_control_enable(I2C0);

i2c_slave_byte_control_disable

The description of i2c_slave_byte_control_disable is shown as below:

Table 3-450. Function i2c_slave_byte_control_disable

Function name	i2c_slave_byte_control_disable
Function prototype	void i2c_slave_byte_control_disable(uint32_t i2c_periph);
Function descriptions	disable slave byte control
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable slave byte control */
```

```
i2c_slave_byte_control_disable(I2C0);
```

i2c_nack_enable

The description of i2c_nack_enable is shown as below:

Table 3-451. Function i2c_nack_enable

Function name	i2c_nack_enable
Function prototype	void i2c_nack_enable(uint32_t i2c_periph);
Function descriptions	generate a NACK in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* generate a NACK in slave mode */
```

i2c_nack_enable(I2C0);

i2c_nack_disable

The description of i2c_nack_disable is shown as below:

Table 3-452. Function i2c_nack_disable

Function name	i2c_nack_disable
Function prototype	void i2c_nack_disable(uint32_t i2c_periph);
Function descriptions	generate a ACK in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* generate a ACK in slave mode */
```

```
i2c_nack_disable(I2C0);
```

i2c_wakeup_from_deepsleep_enable

The description of i2c_wakeup_from_deepsleep_enable is shown as below:

Table 3-453. Function i2c_wakeup_from_deepsleep_enable

Function name	i2c_wakeup_from_deepsleep_enable
Function prototype	void i2c_wakeup_from_deepsleep_enable(uint32_t i2c_periph);
Function descriptions	enable wakeup from Deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable wakeup from Deep-sleep mode */
```

i2c_wakeup_from_deepsleep_enable(I2C0);

i2c_wakeup_from_deepsleep_disable

The description of i2c_wakeup_from_deepsleep_disable is shown as below:

Table 3-454. Function i2c_wakeup_from_deepsleep_disable

Function name	i2c_wakeup_from_deepsleep_disable
Function prototype	void i2c_wakeup_from_deepsleep_disable(uint32_t i2c_periph);
Function descriptions	disable wakeup from Deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable wakeup from Deep-sleep mode */
```

```
i2c_wakeup_from_deepsleep_disable(I2C0);
```

i2c_enable

The description of i2c_enable is shown as below:

Table 3-455. 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
<i>I2Cx</i>	(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 3-456. 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
<i>I2Cx</i>	(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 3-457. 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
<i>I2Cx</i>	(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 3-458. 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
<i>I2Cx</i>	(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 3-459. Function i2c_data_transmit

Function name	i2c_data_transmit
Function prototype	void i2c_data_transmit(uint32_t i2c_periph, uint32_t data);
Function descriptions	I2C transmit data
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(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 3-460. Function i2c_data_receive

Function name	i2c_data_receive
Function prototype	uint32_t i2c_data_receive(uint32_t i2c_periph);
Function descriptions	I2C receive data
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
uint32_t	0x0000..0x00FF

Example:

```
/* I2C0 receive data */
```

```
uint32_t i2c_receiver;
```

```
i2c_receiver = i2c_data_receive(I2C0);
```

i2c_reload_enable

The description of i2c_reload_enable is shown as below:

Table 3-461. Function i2c_reload_enable

Function name	i2c_reload_enable
Function prototype	void i2c_reload_enable(uint32_t i2c_periph);
Function descriptions	enable I2C reload mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C reload mode */
```

```
i2c_reload_enable(I2C0);
```

i2c_reload_disable

The description of i2c_reload_disable is shown as below:

Table 3-462. Function i2c_reload_disable

Function name	i2c_reload_disable
Function prototype	void i2c_reload_disable(uint32_t i2c_periph);
Function descriptions	disable I2C reload mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C reload mode */
```

```
i2c_reload_disable(I2C0);
```

i2c_transfer_byte_number_config

The description of i2c_transfer_byte_number_config is shown as below:

Table 3-463. Function i2c_transfer_byte_number_config

Function name	i2c_transfer_byte_number_config
Function prototype	void i2c_transfer_byte_number_config(uint32_t i2c_periph, uint32_t byte_number);
Function descriptions	configure number of bytes to be transferred
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
byte_number	0x0-0xFF, number of bytes to be transferred
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure number of bytes to be transferred */
```

```
i2c_transfer_byte_number_config(I2C0, 0xFF);
```

i2c_dma_enable

The description of i2c_dma_enable is shown as below:

Table 3-464. Function i2c_dma_enable

Function name	i2c_dma_enable
Function prototype	void i2c_dma_enable(uint32_t i2c_periph, uint8_t dma);
Function descriptions	enable I2C DMA for transmission or reception
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
dma	I2C DMA
<i>I2C_DMA_TRANSMIT</i>	transmit data using DMA
<i>I2C_DMA_RECEIVE</i>	receive data using DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C DMA for transmission or reception */
```

```
i2c_dma_enable(I2C0, I2C_DMA_RECEIVE);
```

i2c_dma_disable

The description of i2c_dma_disable is shown as below:

Table 3-465. Function i2c_dma_disable

Function name	i2c_dma_disable
Function prototype	void i2c_dma_disable(uint32_t i2c_periph, uint8_t dma);
Function descriptions	disable I2C DMA for transmission or reception
Precondition	-
The called functions	-

Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
dma	I2C DMA
<i>I2C_DMA_TRANSMIT</i>	transmit data using DMA
<i>I2C_DMA_RECEIVE</i>	receive data using DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C DMA for transmission or reception */
```

```
i2c_dma_disable(I2C0, I2C_DMA_RECEIVE);
```

i2c_pec_transfer

The description of i2c_pec_transfer is shown as below:

Table 3-466. Function i2c_pec_transfer

Function name	i2c_pec_transfer
Function prototype	void i2c_pec_transfer(uint32_t i2c_periph);
Function descriptions	I2C transfers PEC value
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C transfers PEC value */
```

```
i2c_pec_transfer(I2C0);
```

i2c_pec_enable

The description of i2c_pec_enable is shown as below:

Table 3-467. Function i2c_pec_enable

Function name	i2c_pec_enable
----------------------	----------------

Function prototype	void i2c_pec_enable(uint32_t i2c_periph);
Function descriptions	enable I2C PEC calculation
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C PEC calculation */
i2c_pec_enable(I2C0);
```

i2c_pec_disable

The description of i2c_pec_disable is shown as below:

Table 3-468. Function i2c_pec_disable

Function name	i2c_pec_disable
Function prototype	void i2c_pec_disable(uint32_t i2c_periph);
Function descriptions	disable I2C PEC calculation
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C PEC calculation */
i2c_pec_disable(I2C0);
```

i2c_pec_value_get

The description of i2c_pec_value_get is shown as below:

Table 3-469. Function i2c_pec_value_get

Function name	i2c_pec_value_get
----------------------	-------------------

Function prototype	uint32_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
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
uint32_t	PEC value

Example:

```
/* I2C0 get packet error checking value */
uint32_t pec_value;
pec_value = i2c_pec_value_get(I2C0);
```

i2c_smbus_alert_enable

The description of i2c_smbus_alert_enable is shown as below:

Table 3-470. Function i2c_smbus_alert_enable

Function name	i2c_smbus_alert_enable
Function prototype	void i2c_smbus_alert_enable(uint32_t i2c_periph);
Function descriptions	enable SMBus Alert
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SMBus Alert */
i2c_smbus_alert_enable(I2C0);
```

i2c_smbus_alert_disable

The description of i2c_smbus_alert_disable is shown as below:

Table 3-471. Function i2c_smbus_alert_disable

Function name	i2c_smbus_alert_disable
Function prototype	void i2c_smbus_alert_disable(uint32_t i2c_periph);
Function descriptions	disable SMBus Alert
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SMBus Alert */
i2c_smbus_alert_disable(I2C0);
```

i2c_smbus_default_addr_enable

The description of i2c_smbus_default_addr_enable is shown as below:

Table 3-472. Function i2c_smbus_default_addr_enable

Function name	i2c_smbus_default_addr_enable
Function prototype	void i2c_smbus_default_addr_enable(uint32_t i2c_periph);
Function descriptions	enable SMBus device default address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SMBus device default address */
i2c_smbus_default_addr_enable(I2C0);
```

i2c_smbus_default_addr_disable

The description of i2c_smbus_default_addr_disable is shown as below:

Table 3-473. Function i2c_smbus_default_addr_disable

Function name	i2c_smbus_default_addr_disable
Function prototype	void i2c_smbus_default_addr_disable(uint32_t i2c_periph);
Function descriptions	disable SMBus device default address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SMBus device default address */
```

```
i2c_smbus_default_addr_disable(I2C0);
```

i2c_smbus_host_addr_enable

The description of i2c_smbus_host_addr_enable is shown as below:

Table 3-474. Function i2c_smbus_host_addr_enable

Function name	i2c_smbus_host_addr_enable
Function prototype	void i2c_smbus_host_addr_enable(uint32_t i2c_periph);
Function descriptions	enable SMBus Host address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SMBus Host address */
```

```
i2c_smbus_host_addr_enable(I2C0);
```

i2c_smbus_host_addr_disable

The description of i2c_smbus_host_addr_disable is shown as below:

Table 3-475. Function i2c_smbus_host_addr_disable

Function name	i2c_smbus_host_addr_disable
Function prototype	void i2c_smbus_host_addr_disable(uint32_t i2c_periph);
Function descriptions	disable SMBus Host address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SMBus Host address */
```

```
i2c_smbus_host_addr_disable(I2C0);
```

i2c_extented_clock_timeout_enable

The description of i2c_extented_clock_timeout_enable is shown as below:

Table 3-476. Function i2c_extented_clock_timeout_enable

Function name	i2c_extented_clock_timeout_enable
Function prototype	void i2c_extented_clock_timeout_enable(uint32_t i2c_periph);
Function descriptions	enable extended clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable extended clock timeout detection */
```

```
i2c_extented_clock_timeout_enable(I2C0);
```

i2c_extented_clock_timeout_disable

The description of i2c_extented_clock_timeout_disable is shown as below:

Table 3-477. Function i2c_extented_clock_timeout_disable

Function name	i2c_extented_clock_timeout_disable
Function prototype	void i2c_extented_clock_timeout_disable(uint32_t i2c_periph);
Function descriptions	disable extended clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable extended clock timeout detection */
i2c_extented_clock_timeout_disable(I2C0);
```

i2c_clock_timeout_enable

The description of i2c_clock_timeout_enable is shown as below:

Table 3-478. Function i2c_clock_timeout_enable

Function name	i2c_clock_timeout_enable
Function prototype	void i2c_clock_timeout_enable(uint32_t i2c_periph);
Function descriptions	enable clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable clock timeout detection */
i2c_clock_timeout_enable(I2C0);
```

i2c_clock_timeout_disable

The description of i2c_clock_timeout_disable is shown as below:

Table 3-479. Function i2c_clock_timeout_disable

Function name	i2c_clock_timeout_disable
Function prototype	void i2c_clock_timeout_disable(uint32_t i2c_periph);
Function descriptions	disable clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable clock timeout detection */
i2c_clock_timeout_disable(I2C0);
```

i2c_bus_timeout_b_config

The description of i2c_bus_timeout_b_config is shown as below:

Table 3-480. Function i2c_bus_timeout_b_config

Function name	i2c_bus_timeout_b_config
Function prototype	void i2c_bus_timeout_b_config(uint32_t i2c_periph, uint32_t timeout);
Function descriptions	configure bus timeout B
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
timeout	0-0xffff, bus timeout B
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure bus timeout B */
i2c_bus_timeout_b_config(I2C0, 0xff);
```

i2c_bus_timeout_a_config

The description of i2c_bus_timeout_a_config is shown as below:

Table 3-481. Function i2c_bus_timeout_a_config

Function name	i2c_bus_timeout_a_config
Function prototype	void i2c_bus_timeout_a_config(uint32_t i2c_periph, uint32_t timeout);
Function descriptions	configure bus timeout A
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
timeout	0-0xfff, bus timeout A
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure bus timeout A */
```

```
i2c_bus_timeout_a_config(I2C0, 0xff);
```

i2c_idle_clock_timeout_config

The description of i2c_idle_clock_timeout_config is shown as below:

Table 3-482. Function i2c_idle_clock_timeout_config

Function name	i2c_idle_clock_timeout_config
Function prototype	void i2c_idle_clock_timeout_config(uint32_t i2c_periph, uint32_t timeout);
Function descriptions	configure idle clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
timeout	bus timeout A
<i>BUSTOA_DETECT_SCL_LOW</i>	BUSTOA is used to detect SCL low timeout
<i>BUSTOA_DETECT_IDLE</i>	BUSTOA is used to detect both SCL and SDA high timeout when the bus is idle
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure idle clock timeout detection */
```

```
i2c_idle_clock_timeout_config (I2C0, BUSTOA_DETECT_SCL_LOW);
```

i2c_flag_get

The description of i2c_flag_get is shown as below:

Table 3-483. Function i2c_flag_get

Function name	i2c_flag_get
Function prototype	FlagStatus i2c_flag_get(uint32_t i2c_periph, uint32_t flag);
Function descriptions	get I2C flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1)
Input parameter{in}	
flag	I2C flags
I2C_FLAG_TBE	I2C_TDATA is empty during transmitting
I2C_FLAG_TI	transmit interrupt
I2C_FLAG_RBNE	I2C_RDATA is not empty during receiving
I2C_FLAG_ADDSEND	address received matches in slave mode
I2C_FLAG_NACK	not acknowledge flag
I2C_FLAG_STPDET	STOP condition detected in slave mode
I2C_FLAG_TC	transfer complete in master mode
I2C_FLAG_TCR	transfer complete reload
I2C_FLAG_BERR	bus error
I2C_FLAG_LOSTARB	arbitration Lost
I2C_FLAG_OUERR	overflow/underrun error in slave mode
I2C_FLAG_PECERR	PEC error
I2C_FLAG_TIMEOUT	timeout flag
I2C_FLAG_SMBALT	SMBus Alert
I2C_FLAG_I2CBSY	busy flag
I2C_FLAG_TR	whether the I2C is a transmitter or a receiver in slave mode
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get I2C flag status */

FlagStatus flag_state = RESET;

flag_state = i2c_flag_get (I2C0, I2C_FLAG_TBE);
```

i2c_flag_clear

The description of i2c_flag_clear is shown as below:

Table 3-484. Function i2c_flag_clear

Function name	i2c_flag_clear
Function prototype	void i2c_flag_clear(uint32_t i2c_periph, uint32_t 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	I2C flags
I2C_FLAG_ADDSEND	address received matches in slave mode
I2C_FLAG_NACK	not acknowledge flag
I2C_FLAG_STPDET	STOP condition detected in slave mode
I2C_FLAG_BERR	bus error
I2C_FLAG_LOSTARB	arbitration Lost
I2C_FLAG_OUERR	overflow/underrun error in slave mode
I2C_FLAG_PECERR	PEC error
I2C_FLAG_TIMEOUT	timeout flag
I2C_FLAG_SMBALT	SMBus Alert
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 3-485. Function `i2c_interrupt_enable`

Function name	<code>i2c_interrupt_enable</code>
Function prototype	<code>void i2c_interrupt_enable(uint32_t i2c_periph, uint32_t interrupt);</code>
Function descriptions	enable I2C interrupt
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
interrupt	I2C interrupts
<i>I2C_INT_ERR</i>	error interrupt
<i>I2C_INT_TC</i>	transfer complete interrupt
<i>I2C_INT_STPDET</i>	stop detection interrupt
<i>I2C_INT_NACK</i>	not acknowledge received interrupt
<i>I2C_INT_ADDM</i>	address match interrupt
<i>I2C_INT_RBNE</i>	receive interrupt
<i>I2C_INT_TI</i>	transmit interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C0 transmit interrupt */
i2c_interrupt_enable(I2C0, I2C_INT_TI);
```

`i2c_interrupt_disable`

The description of `i2c_interrupt_disable` is shown as below:

Table 3-486. Function `i2c_interrupt_disable`

Function name	<code>i2c_interrupt_disable</code>
Function prototype	<code>void i2c_interrupt_disable(uint32_t i2c_periph, uint32_t interrupt);</code>
Function descriptions	disable I2C interrupt
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
interrupt	I2C interrupts
<i>I2C_INT_ERR</i>	error interrupt

<i>I2C_INT_TC</i>	transfer complete interrupt
<i>I2C_INT_STPDET</i>	stop detection interrupt
<i>I2C_INT_NACK</i>	not acknowledge received interrupt
<i>I2C_INT_ADDM</i>	address match interrupt
<i>I2C_INT_RBNE</i>	receive interrupt
<i>I2C_INT_TI</i>	transmit interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C0 transmit interrupt */
```

```
i2c_interrupt_disable(I2C0, I2C_INT_TI);
```

i2c_interrupt_flag_get

The description of i2c_interrupt_flag_get is shown as below:

Table 3-487. 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 status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
int_flag	I2C interrupt flags, refer to Table 3-425. i2c_interrupt_flag_enum .
<i>I2C_INT_FLAG_TI</i>	transmit interrupt flag
<i>I2C_INT_FLAG_RBNE</i>	I2C_RDATA is not empty during receiving interrupt flag
<i>I2C_INT_FLAG_ADDS</i> <i>END</i>	address received matches in slave mode interrupt flag
<i>I2C_INT_FLAG_NACK</i>	not acknowledge interrupt flag
<i>I2C_INT_FLAG_STPD</i> <i>ET</i>	stop condition detected in slave mode interrupt flag
<i>I2C_INT_FLAG_TC</i>	transfer complete in master mode interrupt flag
<i>I2C_INT_FLAG_TCR</i>	transfer complete reload interrupt flag
<i>I2C_INT_FLAG_BERR</i>	bus error interrupt flag
<i>I2C_INT_FLAG_LOSTA</i> <i>RB</i>	arbitration lost interrupt flag

<i>I2C_INT_FLAG_OUER</i> <i>R</i>	overrun/underrun error in slave mode interrupt flag
<i>I2C_INT_FLAG_PECE</i> <i>RR</i>	PEC error interrupt flag
<i>I2C_INT_FLAG_TIMEO</i> <i>UT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBA</i> <i>LT</i>	SMBus Alert interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get I2C interrupt flag status */
```

```
FlagStatus flag_state = RESET;
```

```
flag_state = i2c_interrupt_flag_get(I2C0, I2C_INT_FLAG_TI);
```

i2c_interrupt_flag_clear

The description of i2c_interrupt_flag_clear is shown as below:

Table 3-488. 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
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
int_flag	I2C interrupt flags, refer to Table 3-425. i2c_interrupt_flag_enum.
<i>I2C_INT_FLAG_ADDS</i> <i>END</i>	address received matches in slave mode interrupt flag
<i>I2C_INT_FLAG_NACK</i>	not acknowledge interrupt flag
<i>I2C_INT_FLAG_STPD</i> <i>ET</i>	stop condition detected in slave mode interrupt flag
<i>I2C_INT_FLAG_BERR</i>	bus error interrupt flag
<i>I2C_INT_FLAG_LOSTA</i> <i>RB</i>	arbitration lost interrupt flag
<i>I2C_INT_FLAG_OUER</i>	overrun/underrun error in slave mode interrupt flag

<i>R</i>	
<i>I2C_INT_FLAG_PECERR</i>	PEC error interrupt flag
<i>I2C_INT_FLAG_TIMEOUT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBALERT</i>	SMBus Alert interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear a bus error flag */
i2c_interrupt_flag_clear(I2C0, I2C_INT_FLAG_BERR);
```

3.16. ICACHE

The instruction cache (ICACHE) is based on C-AHB code bus of Cortex-M33 processor. It is necessary to improve performance in fetching instruction and data from both internal and external memories. The ICACHE registers are listed in chapter [3.16.1](#), the ICACHE firmware functions are introduced in chapter [3.16.2](#).

3.16.1. Descriptions of Peripheral registers

ICACHE registers are listed in the table shown as below:

Table 3-489. ICACHE Registers

Registers	Descriptions
ICACHE_CTL	ICACHE control register
ICACHE_STAT	ICACHE status register
ICACHE_INTEN	ICACHE interrupt enable register
ICACHE_FC	ICACHE flag clear register
ICACHE_HMC	ICACHE hit monitor counter register
ICACHE_MMC	ICACHE miss monitor counter register
ICACHE_CFGx	ICACHE configuration register

3.16.2. Descriptions of Peripheral functions

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

Table 3-490. ICACHE firmware function

Function name	Function description
icache_enable	enable icache
icache_disable	disable icache
icache_monitor_enable	enable the icache monitor
icache_monitor_disable	disable the icache monitor
icache_monitor_reset	reset the icache monitor
icache_way_configure	configure icache way (associativity mode)
icache_burst_type_select	select icache burst type
icache_invalidation	invalidate icache
icache_hitvalue_get	get the hit monitor value
icache_missvalue_get	get the miss monitor value
icache_remap_enable	enable the icache remap function
icache_remap_disable	disable the icache remap function
icache_flag_get	get icache flag
icache_flag_clear	clear icache flag
icache_interrupt_enable	enable icache interrupt
icache_interrupt_disable	disable icache interrupt
icache_interrupt_flag_get	get icache interrupt flag
icache_interrupt_flag_clear	clear icache interrupt flag

Structure icache_remap_struct

Table 3-491. Structure icache_remap_struct

Member name	Function description
region_num	remap region number
base_address	remap base address
remap_address	remap address
remap_size	remap size
burst_type	remap burst type
master_sel	remap master selection

icache_enable

The description of icache_enable is shown as below:

Table 3-492. Function icache_enable

Function name	icache_enable
Function prototype	void icache_enable(void);
Function descriptions	enable icache
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable icache */
```

```
icache_enable ();
```

icache_disable

The description of icache_disable is shown as below:

Table 3-493. Function icache_disable

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

Example:

```
/* disable icache */
```

```
icache_disable ();
```

icache_monitor_enable

The description of icache_monitor_enable is shown as below:

Table 3-494. Function icache_monitor_enable

Function name	icache_monitor_enable
Function prototype	void icache_monitor_enable(uint32_t monitor_source);
Function descriptions	enable the icache monitor
Precondition	-
The called functions	-
Input parameter{in}	
monitor_source	the monitor to be enabled
ICACHE_MONITOR_HIT	hit monitor

ICACHE_MONITOR_MISS	miss monitor
ICACHE_MONITOR_HIT_MISS	hit and miss monitor
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the icache monitor */
```

```
icache_monitor_enable (ICACHE_MONITOR_HIT);
```

icache_monitor_disable

The description of icache_monitor_disable is shown as below:

Table 3-495. Function icache_monitor_disable

Function name	icache_monitor_disable
Function prototype	void icache_monitor_disable(uint32_t monitor_source);
Function descriptions	disable the icache monitor
Precondition	-
The called functions	-
Input parameter{in}	
monitor_source	the monitor to be disabled
ICACHE_MONITOR_HIT	hit monitor
ICACHE_MONITOR_MISS	miss monitor
ICACHE_MONITOR_HIT_MISS	hit and miss monitor
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the icache monitor */
```

```
icache_monitor_disable (ICACHE_MONITOR_HIT);
```

icache_monitor_reset

The description of icache_monitor_reset is shown as below:

Table 3-496. Function icache_monitor_reset

Function name	icache_monitor_reset
Function prototype	void icache_monitor_reset(uint32_t reset_monitor_source);
Function descriptions	reset the icache monitor
Precondition	-
The called functions	-
Input parameter{in}	
reset_monitor_source	the monitor to be reset
ICACHE_MONITOR_HIT	hit monitor
ICACHE_MONITOR_MISS	miss monitor
ICACHE_MONITOR_HIT_MISS	hit and miss monitor
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the icache monitor */
```

```
icache_monitor_reset (ICACHE_MONITOR_HIT);
```

icache_way_configure

The description of icache_way_configure is shown as below:

Table 3-497. Function icache_way_configure

Function name	icache_way_configure
Function prototype	ErrStatus icache_way_configure(void);
Function descriptions	Configure icache way (associativity mode)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* Configure icache way */
```

```
ErrStatus Flag = icache_way_configure ();
```

icache_burst_type_select

The description of icache_burst_type_select is shown as below:

Table 3-498. Function icache_burst_type_select

Function name	icache_burst_type_select
Function prototype	ErrStatus icache_burst_type_select(uint32_t burst_type);
Function descriptions	select icache burst type
Precondition	-
The called functions	-
Input parameter{in}	
burst_type	output burst type
ICACHE_WRAP_BURST	icache WRAP burst mode
ICACHE_INCR_BURST	icache INCR burst mode
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* select icache burst type */
```

```
ErrStatus Flag = icache_burst_type_select (ICACHE_WRAP_BURST);
```

icache_invalidation

The description of icache_invalidation is shown as below:

Table 3-499. Function icache_invalidation

Function name	icache_invalidation
Function prototype	ErrStatus icache_invalidation(void);
Function descriptions	invalidate icache
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* invalid icache */
```

ErrStatus Flag = icache_invalidation();

icache_hitvalue_get

The description of icache_hitvalue_get is shown as below:

Table 3-500. Function icache_hitvalue_get

Function name	icache_hitvalue_get
Function prototype	uint32_t icache_hitvalue_get(void);
Function descriptions	get the hit monitor value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit value of the hit monitor counter register (0-0Xffff ffff)

Example:

```
/* get the hit monitor value */
uint8_t hit_value = 0;
hit_value = icache_hitvalue_get ();
```

icache_missvalue_get

The description of icache_missvalue_get is shown as below:

Table 3-501. Function icache_missvalue_get

Function name	icache_missvalue_get
Function prototype	uint32_t icache_missvalue_get(void);
Function descriptions	get the miss monitor value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	16-bit value of the miss monitor counter register (0-0Xffff)

Example:

```
/* get the miss monitor value */
```

```
uint8_t miss_value = 0;
```

```
miss_value = icache_missvalue_get ();
```

icache_remap_enable

The description of icache_remap_enable is shown as below:

Table 3-502. Function icache_remap_enable

Function name	icache_remap_enable
Function prototype	ErrStatus icache_remap_enable(icache_remap_struct * icache_remap_config);
Function descriptions	enable the icache remap function
Precondition	-
The called functions	-
Input parameter{in}	
icache_remap_config	icache remap structure, refer to Table 3-491. Structure icache_remap_struct
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* enable the icache remap function */
icache_remap_struct ICACHE_struct;
ICACHE_struct.region_num = 0U;
ICACHE_struct.base_address = 0x10000000UL;
ICACHE_struct.remap_address = 0x30000000UL;
ICACHE_struct.remap_size = ICACHE_REMAP_2M;
ICACHE_struct.burst_type = ICACHE_WRAP_BURST;
icache_remap_enable(&ICACHE_struct);
```

icache_remap_disable

The description of icache_remap_disable is shown as below:

Table 3-503. Function icache_remap_disable

Function name	icache_remap_disable
Function prototype	ErrStatus icache_remap_disable(uint32_t region_num);
Function descriptions	disable the icache remap function
Precondition	-

The called functions	-
Input parameter{in}	
region_num	the remap region to be disabled
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* disable the icache remap function */
```

```
icache_remap_disable (1);
```

icache_flag_get

The description of icache_flag_get is shown as below:

Table 3-504. Function icache_flag_get

Function name	icache_flag_get
Function prototype	FlagStatus icache_flag_get(uint32_t flag);
Function descriptions	get icache flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	the icache flag to be get
ICACHE_BUSY_FLAG	icache busy flag
ICACHE_END_FLAG	icache busy end flag
ICACHE_ERR_FLAG	icache error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get icache flag */
```

```
FlagStatus flag_value;
```

```
flag_value = icache_flag_get (ICACHE_BUSY_FLAG);
```

icache_flag_clear

The description of icache_flag_clear is shown as below:

Table 3-505. Function icache_flag_clear

Function name	icache_flag_clear
----------------------	-------------------

Function prototype	void icache_flag_clear(uint32_t flag);
Function descriptions	clear icache flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	the flag to be cleared
<i>ICACHE_ENDC_FLAG</i>	icache busy end clear flag
<i>ICACHE_ERRC_FLAG</i>	icache error clear flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear icache flag */
icache_flag_clear (ICACHE_ENDC_FLAG);
```

icache_interrupt_enable

The description of icache_interrupt_enable is shown as below:

Table 3-506. Function icache_interrupt_enable

Function name	icache_interrupt_enable
Function prototype	void icache_interrupt_enable(uint32_t interrupt);
Function descriptions	enable icache interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the interrupt to be enabled
<i>ICACHE_ENDIE</i> :	icache busy end interrupt
<i>ICACHE_ERRIE</i>	icache error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable icache interrupt */
icache_interrupt_enable (ICACHE_ENDIE);
```

icache_interrupt_disable

The description of icache_interrupt_disable is shown as below:

Table 3-507. Function `icache_interrupt_disable`

Function name	<code>icache_interrupt_disable</code>
Function prototype	<code>void icache_interrupt_disable(uint32_t interrupt);</code>
Function descriptions	disable icache interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the interrupt to be disabled
<code>ICACHE_ENDIE</code>	icache busy end interrupt
<code>ICACHE_ERRIE</code>	icache error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable icache interrupt */
icache_interrupt_disable (ICACHE_ENDIE);
```

`icache_interrupt_flag_get`

The description of `icache_interrupt_flag_get` is shown as below:

Table 3-508. Function `icache_flag_get`

Function name	<code>icache_interrupt_flag_get</code>
Function prototype	<code>FlagStatus icache_interrupt_flag_get(uint32_t interrupt);</code>
Function descriptions	get icache interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the icache interrupt flag to be get
<code>ICACHE_END_FLAG</code>	icache busy end interrupt flag
<code>ICACHE_ERR_FLAG</code>	icache error interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get icache flag */
FlagStatus flag_value;

flag_value = icache_interrupt_flag_get (ICACHE_END_FLAG);
```

icache_interrupt_flag_clear

The description of icache_interrupt_flag_clear is shown as below:

Table 3-509. Function icache_interrupt_flag_clear

Function name	icache_interrupt_flag_clear
Function prototype	void icache_interrupt_flag_clear(uint32_t interrupt);
Function descriptions	clear icache interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the interrupt flag to clear
ICACHE_ENDC_FLAG	icache busy end clear interrupt flag
ICACHE_ERRC_FLAG	icache error clear interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear icache interrupt flag */
```

```
icache_interrupt_flag_clear (ICACHE_ENDC_FLAG);
```

3.17. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick. The NVIC and SysTick registers are listed in chapter [3.17.1](#), the MISC firmware functions are introduced in chapter [3.17.2](#).

3.17.1. Descriptions of Peripheral registers

Table 3-510. 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
ITNS ⁽¹⁾	Interrupt Non-Secure State Register
IPR ⁽¹⁾	Interrupt Priority Register
STIR ⁽¹⁾	Software Trigger Interrupt Register
CPUID ⁽²⁾	CPUID Base Register

Registers	Descriptions
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
SHPR ⁽²⁾	System Handlers Priority Registers
SHCSR ⁽²⁾	System Handler Control and State Register

1. refer to the structure NVIC_Type, is defined in the core_cm33.h file

2. refer to the structure SCB_Type, is defined in the core_cm33.h file

Table 3-511. 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_cm33.h file

3.17.2. Descriptions of Peripheral functions

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

Table 3-512. 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_system_reset	initiates a system reset request to reset the MCU
nvic_vector_table_set	set the NVIC vector table 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

Enum IRQn_Type

Table 3-513. IRQn_Type

Member name	Function description
WWDGT_IRQn	Window watchdog timer interrupt
LVD_IRQn	LVD through EXTI Line detection interrupt
TAMPER_STAMP_IRQn	RTC Tamper and TimeStamp events interrupt
RTC_WKUP_IRQn	RTC Wakeup event interrupt
FMC_IRQn	FMC global interrupt
RCU_IRQn	RCU global interrupt

Member name	Function description
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
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_Channel6_IRQn	DMA0 channel6 global interrupt
DMA0_Channel7_IRQn	DMA0 channel7 global interrupt
ADC_IRQn	ADC interrupt
TAMPER_STAMP_S_IRQn	RTC Tamper and TimeStamp events security interrupt
RTC_WKUP_S_IRQn	RTC Wakeup event security interrupt
RTC_Alarm_S_IRQn	RTC Alarm event security interrupt
EXTI5_9_IRQn	EXTI Line5-9 interrupt
TIMER0_BRK_IRQn	TIMER0 Break interrupt
TIMER0_UP_IRQn	TIMER0 update interrupt
TIMER0_CMT_IRQn	TIMER0 commutation interrupt
TIMER0_Channel_IRQn	TIMER0 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/I2S1 global interrupt
USART0_IRQn	USART0 global interrupt
USART1_IRQn	USART1 global interrupt
USART2_IRQn	USART2 global interrupt
EXTI10_15_IRQn	EXTI Line10-15 interrupt
RTC_Alarm_IRQn	RTC Alarm event interrupt
VLVDF_IRQn	VLVDF interrupt
TIMER15_IRQn	TIMER15 global interrupt
TIMER16_IRQn	TIMER16 global interrupt
SDIO_IRQn	SDIO global interrupt
TIMER4_IRQn	TIMER4 global interrupt

Member name	Function description
I2C0_WKUP_IRQn	I2C0 Wakeup interrupt
USART0_IRQn	USART0 Wakeup
USART2_IRQn	USART2 Wakeup
TIMER5_IRQn	TIMER5 global interrupt
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
DMA1_Channel5_IRQn	DMA1 channel5 global interrupt
DMA1_Channel6_IRQn	DMA1 channel6 global interrupt
DMA1_Channel7_IRQn	DMA1 channel7 global interrupt
WIFI11N_WKUP_IRQn	WIFI11N wakeup interrupt
USBFS_IRQn	USBFS global interrupt
USBFS_WKUP_IRQn	USBFS wakeup interrupt
DCI_IRQn	DCI global interrupt
CAU_IRQn	CAU global interrupt
HAU_TRNG_IRQn	HAU/TRNG global interrupt
FPU_IRQn	FPU global interrupt
HPDF_INT0_IRQn	HPDF global interrupt0
HPDF_INT1_IRQn	HPDF global interrupt1
WIFI11N_INT0_IRQn	WIFI11N global interrupt0
WIFI11N_INT1_IRQn	WIFI11N global interrupt1
WIFI11N_INT2_IRQn	WIFI11N global interrupt2
EFUSE_IRQn	EFUSE global interrupt
QSPI_IRQn	QSPI global interrupt
PKCAU_IRQn	PKCAU global interrupt
TSI_IRQ	TSI global interrupt
ICACHE_IRQn	ICACHE global interrupt
TZIAC_S_IRQn	TZIAC security interrupt
FMC_S_IRQn	FMC secure interrupt
QSPI_S_IRQn	QSPI security interrupt

nvic_priority_group_set

The description of nvic_priority_group_set is shown as below:

Table 3-514. Function nvic_priority_group_set

Function name	nvic_priority_group_set
Function prototype	void nvic_priority_group_set(uint32_t nvic_prigroup);
Function descriptions	configure bits length of the priority group
Precondition	-

The called functions	-
Input parameter{in}	
nvic_prigroup	priority group
<i>NVIC_PRIGROUP_PRE0_SUB4</i>	0 bits for pre-emption priority 4 bits for subpriority
<i>NVIC_PRIGROUP_PRE1_SUB3</i>	1 bits for pre-emption priority 3 bits for subpriority
<i>NVIC_PRIGROUP_PRE2_SUB2</i>	2 bits for pre-emption priority 2 bits for subpriority
<i>NVIC_PRIGROUP_PRE3_SUB1</i>	3 bits for pre-emption priority 1 bits for subpriority
<i>NVIC_PRIGROUP_PRE4_SUB0</i>	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 3-515. 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 request, configure the priority of interrupt
Precondition	-
The called functions	nvic_priority_group_set
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to Table 3-513. IRQn_Type
Input parameter{in}	
nvic_irq_pre_priority	the pre-emption priority needed to set (0~4)
Input parameter{in}	
nvic_irq_sub_priority	the subpriority needed to set (0~4)
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 3-516. Function nvic_irq_disable

Function name	nvic_irq_disable
Function prototype	void nvic_irq_disable(uint8_t nvic_irq);
Function descriptions	disable NVIC request
Precondition	-
The called functions	-
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to Table 3-513. IRQn_Type
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

nvic_system_reset

The description of nvic_system_reset is shown as below:

Table 3-517. Function nvic_system_reset

Function name	nvic_system_reset
Function prototype	void nvic_system_reset(void);
Function descriptions	initiates a system reset request to reset the MCU
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initiates a system reset request to reset the MCU */
```



```
nvic_system_reset();
```

nvic_vector_table_set

The description of nvic_vector_table_set is shown as below:

Table 3-518. Function nvic_vector_table_set

Function name	nvic_vector_table_set
Function prototype	void nvic_vector_table_set(uint32_t nvic_vect_tab, uint32_t offset);
Function descriptions	set the NVIC vector table address
Precondition	-
The called functions	-
Input parameter{in}	
nvic_vect_tab	the RAM or FLASH base address
<i>NVIC_VECTTAB_RAM</i>	RAM base address
<i>NVIC_VECTTAB_FLASH</i>	Flash base address
<i>NVIC_VECTTAB_RAM_SECURE</i>	RAM base address of secure world
<i>NVIC_VECTTAB_FLASH_SECURE</i>	Flash base address of secure world
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 3-519. Function system_lowpower_set

Function name	system_lowpower_set
Function prototype	void system_lowpower_set(uint8_t lowpower_mode);
Function descriptions	the state of the low power mode management
Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state

<i>SCB_LPM_SLEEP_EXIT_ISR</i>	if chose this para, the system always enter low power mode by exiting from ISR
<i>SCB_LPM_DEEPSLEEP</i>	if chose this para, the system will enter the DEEPSLEEP mode
<i>SCB_LPM_WAKE_BY_ALL_INT</i>	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 3-520. Function system_lowpower_reset

Function name	system_lowpower_reset
Function prototype	void system_lowpower_reset(uint8_t lowpower_mode);
Function descriptions	the state of the low power mode management
Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state
<i>SCB_LPM_SLEEP_EXIT_ISR</i>	if chose this para, the system will exit low power mode by exiting from ISR
<i>SCB_LPM_DEEPSLEEP</i>	if chose this para, the system will enter the SLEEP mode
<i>SCB_LPM_WAKE_BY_ALL_INT</i>	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 3-521. 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_CLKSOURC E_HCLK	systick clock source is from HCLK
SYSTICK_CLKSOURC E_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);
```

3.18. PKCAU

The Public Key Cryptographic Acceleration Unit (PKCAU) can accelerate RSA (Rivest, Shamir and Adleman), Diffie-Hellmann (DH key exchange) and ECC (elliptic curve cryptography) in GF(p) (Galois domain). The PKCAU registers are listed in chapter [3.18.1](#), the PKCAU firmware functions are introduced in chapter [3.18.2](#).

3.18.1. Descriptions of Peripheral registers

PKCAU registers are listed in the table shown as below:

Table 3-522. PKCAU Registers

Registers	Descriptions
PKCAU_CTL	Control register
PKCAU_STAT	Status register
PKCAU_STATC	Status clear register

3.18.2. Descriptions of Peripheral functions

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

Table 3-523. PKCAU firmware function

Function name	Function description
pkcau_deinit	reset PKCAU
pkcau_mont_struct_para_init	initialize montgomery parameter structure with a default value
pkcau_mod_struct_para_init	initialize modular parameter structure with a default value
pkcau_mod_exp_struct_para_init	initialize modular exponentation parameter structure with a default value
pkcau_mod_inver_struct_para_init	initialize modular inversion parameter structure with a default value
pkcau_arithmetic_struct_para_init	initialize arithmetic parameter structure with a default value
pkcau_crt_struct_para_init	initialize CRT parameter structure with a default value
pkcau_ec_group_struct_para_init	initialize ECC curve parameter structure with a default value
pkcau_point_struct_para_init	initialize point parameter structure with a default value
pkcau_signature_struct_para_init	initialize signature parameter structure with a default value
pkcau_hash_struct_para_init	initialize hash parameter structure with a default value
pkcau_mod_reduc_struct_para_init	initialize modular reduction parameter structure with a default value
pkcau_enable	enable PKCAU
pkcau_disable	disable PKCAU
pkcau_start	start operation
pkcau_mode_set	configure the PKCAU operation mode
pkcau_mont_param_operation	execute montgomery parameter operation
pkcau_mod_operation	execute modular operation, include modular addition, modular subtraction and montgomery multiplication
pkcau_mod_exp_operation	execute modular exponentation operation
pkcau_mod_inver_operation	execute modular inversion operation
pkcau_mod_reduc_operation	execute modular reduction operation
pkcau_arithmetic_operation	execute arithmetic addition operation
pkcau_crt_exp_operation	execute RSA CRT exponentation operation
pkcau_point_check_operation	execute point check operation
pkcau_point_mul_operation	execute point multiplication operation
pkcau_ecdsa_sign_operation	execute ECDSA sign operation
pkcau_ecdsa_verification_operation	execute ECDSA verify operation
pkcau_memread	read result from PKCAU RAM
pkcau_flag_get	get PKCAU flag status
pkcau_flag_clear	clear PKCAU flag status
pkcau_interrupt_enable	enable PKCAU interrupt
pkcau_interrupt_disable	disable PKCAU interrupt
pkcau_interrupt_flag_get	get PKCAU interrupt flag status

Function name	Function description
pkcau_interrupt_flag_clear	clear PKCAU interrupt flag status

Structure pkcau_mont_parameter_struct

Table 3-524. Structure pkcau_mont_parameter_struct

Member name	Function description
modulus_n	modulus value n
modulus_len	modulus length in byte

Structure pkcau_mod_parameter_struct

Table 3-525. Structure pkcau_mod_parameter_struct

Member name	Function description
opr_d_a	operand A
opr_d_b	operand B
modulus_n	modulus value n
modulus_len	modulus length in byte

Structure pkcau_mod_exp_parameter_struct

Table 3-526. Structure pkcau_mod_exp_parameter_struct

Member name	Function description
opr_d_a	operand A
exp_e	exponent e
e_len	exponent length in byte
modulus_n	modulus n
modulus_len	modulus length in byte
mont_para	montgomery parameter R2 mod n

Structure pkcau_mod_inver_parameter_struct

Table 3-527. Structure pkcau_mod_inver_parameter_struct

Member name	Function description
opr_d_a	operand A
modulus_n	modulus value n
modulus_len	modulus length in byte

Structure pkcau_mod_reduc_parameter_struct

Table 3-528. Structure pkcau_mod_reduc_parameter_struct

Member name	Function description
opr_d_a	operand A
opr_d_a_len	length of operand A in byte
modulus_n	modulus value n

Member name	Function description
modulus_len	modulus length in byte

Structure pkcau_arithmetic_parameter_struct

Table 3-529. Structure pkcau_arithmetic_parameter_struct

Member name	Function description
opr_d_a	operand A
opr_d_b	operand B
opr_d_len	length of operand in byte

Structure pkcau_crt_parameter_struct

Table 3-530. Structure pkcau_crt_parameter_struct

Member name	Function description
opr_d_a	operand A
opr_d_len	length of operand in byte
opr_d_dp	operand dp
opr_d_dq	operand dq
opr_d_qinv	operand qinv
opr_d_p	prime operand p
opr_d_q	prime operand q

Structure pkcau_ec_group_parameter_struct

Table 3-531. Structure pkcau_ec_group_parameter_struct

Member name	Function description
modulus_p	curve modulus p
coff_a	curve coefficient a
coff_b	curve coefficient b
base_point_x	curve base point coordinate x
base_point_y	curve base point coordinate y
order_n	curve prime order n
a_sign	curve coefficient a sign
modulus_p_len	curve modulus p length in byte
order_n_len	curve prime order n length in byte
multi_k	scalar multiplier k
k_len	length of scalar multiplier k
mont_para	montgomery parameter R2 mod n

Structure pkcau_point_parameter_struct

Table 3-532. Structure pkcau_point_parameter_struct

Member name	Function description
point_x	point coordinate x
point_y	point coordinate y

Structure pkcau_signature_parameter_struct

Table 3-533. Structure pkcau_signature_parameter_struct

Member name	Function description
sign_r	signature part r
sign_s	signature part s

Structure pkcau_hash_parameter_struct

Table 3-534. Structure pkcau_hash_parameter_struct

Member name	Function description
hash_z	hash value z
hash_z_len	hash value z length in byte

pkcau_deinit

The description of pkcau_deinit is shown as below:

Table 3-535. Function pkcau_deinit

Function name	pkcau_deinit
Function prototype	void pkcau_deinit(void);
Function descriptions	reset PKCAU
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset PKCAU */
pkcau_deinit();
```

pkcau_mont_struct_para_init

The description of pkcau_mont_struct_para_init is shown as below:

Table 3-536. Function pkcau_mont_struct_para_init

Function name	pkcau_mont_struct_para_init
Function prototype	void pkcau_mont_struct_para_init(pkcau_mont_parameter_struct* init_para);
Function descriptions	initialize montgomery parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	montgomery parameter struct, the structure members can refer to Table 3-524. Structure pkcau_mont_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU montgomery parameter struct with a default value */
```

```
pkcau_mont_parameter_struct pkcau_mont_parameter;
```

```
pkcau_mont_struct_para_init(&pkcau_mont_parameter);
```

pkcau_mod_struct_para_init

The description of pkcau_mod_struct_para_init is shown as below:

Table 3-537. Function pkcau_mod_struct_para_init

Function name	pkcau_mod_struct_para_init
Function prototype	void pkcau_mod_struct_para_init(pkcau_mod_parameter_struct* init_para);
Function descriptions	initialize modular parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	modular parameter struct, the structure members can refer to Table 3-525. Structure pkcau_mod_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU modular parameter struct with a default value */
```

```
pkcau_mod_parameter_struct pkcau_mod_parameter;
```

```
pkcau_mod_struct_para_init(&pkcau_mod_parameter);
```


pkcau_mod_exp_struct_para_init

The description of pkcau_mod_exp_struct_para_init is shown as below:

Table 3-538. Function pkcau_mod_exp_struct_para_init

Function name	pkcau_mod_exp_struct_para_init
Function prototype	void pkcau_mod_exp_struct_para_init(pkcau_mod_exp_parameter_struct* init_para);
Function descriptions	initialize modular exponentation parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	modular exponentation parameter struct, the structure members can refer to Table 3-526. Structure pkcau_mod_exp_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU modular exponentation parameter struct with a default value */
```

```
pkcau_mod_exp_parameter_struct pkcau_mod_exp_parameter;
```

```
pkcau_mod_exp_struct_para_init(&pkcau_mod_exp_parameter);
```

pkcau_mod_inver_struct_para_init

The description of pkcau_mod_inver_struct_para_init is shown as below:

Table 3-539. Function pkcau_mod_inver_struct_para_init

Function name	pkcau_mod_inver_struct_para_init
Function prototype	void pkcau_mod_inver_struct_para_init(pkcau_mod_inver_parameter_struct* init_para);
Function descriptions	initialize modular inversion parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	modular inversion parameter struct, the structure members can refer to Table 3-527. Structure pkcau_mod_inver_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU modular inversion parameter struct with a default value */

pkcau_mod_inver_parameter_struct pkcau_mod_inver_parameter;

pkcau_mod_inver_struct_para_init(&pkcau_mod_inver_parameter);
```

pkcau_arithmetic_struct_para_init

The description of pkcau_arithmetic_struct_para_init is shown as below:

Table 3-540. Function pkcau_arithmetic_struct_para_init

Function name	pkcau_arithmetic_struct_para_init
Function prototype	void pkcau_arithmetic_struct_para_init(pkcau_arithmetic_parameter_struct* init_para);
Function descriptions	initialize arithmetic parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	arithmetic parameter struct, the structure members can refer to Table 3-529. Structure pkcau_arithmetic_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU arithmetic parameter struct struct with a default value */

pkcau_arithmetic_parameter_struct pkcau_arithmetic_parameter;

pkcau_arithmetic_struct_para_init(&pkcau_arithmetic_parameter);
```

pkcau crt_struct_para_init

The description of pkcau crt_struct_para_init is shown as below:

Table 3-541. Function pkcau crt_struct_para_init

Function name	pkcau crt_struct_para_init
Function prototype	void pkcau crt_struct_para_init(pkcau crt_parameter_struct* init_para);
Function descriptions	initialize CRT parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	CRT parameter struct, the structure members can refer to Table 3-530. Structure pkcau crt parameter struct.

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU CRT parameter struct with a default value */
```

```
pkcau_crt_parameter_struct pkcau_crt_parameter;
```

```
pkcau_crt_struct_para_init(&pkcau_crt_parameter);
```

pkcau_ec_group_struct_para_init

The description of pkcau_ec_group_struct_para_init is shown as below:

Table 3-542. Function pkcau_ec_group_struct_para_init

Function name	pkcau_ec_group_struct_para_init
Function prototype	void pkcau_ec_group_struct_para_init(pkcau_ec_group_parameter_struct* init_para);
Function descriptions	initialize ECC curve parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	ECC curve parameter struct, the structure members can refer to Table 3-531. Structure pkcau_ec_group_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU ECC curve parameter struct with a default value */
```

```
pkcau_ec_group_parameter_struct pkcau_ec_group_parameter;
```

```
pkcau_ec_group_struct_para_init(&pkcau_ec_group_parameter);
```

pkcau_point_struct_para_init

The description of pkcau_point_struct_para_init is shown as below:

Table 3-543. Function pkcau_point_struct_para_init

Function name	pkcau_point_struct_para_init
Function prototype	void pkcau_point_struct_para_init(pkcau_point_parameter_struct* init_para);
Function descriptions	initialize point parameter structure with a default value

Precondition	-
The called functions	-
Input parameter{in}	
init_para	point parameter struct, the structure members can refer to Table 3-532. Structure pkcau_point_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU point parameter struct with a default value */
pkcau_point_parameter_struct pkcau_point_parameter;
pkcau_point_struct_para_init(&pkcau_point_parameter);
```

pkcau_signature_struct_para_init

The description of pkcau_signature_struct_para_init is shown as below:

Table 3-544. Function pkcau_signature_struct_para_init

Function name	pkcau_signature_struct_para_init
Function prototype	void pkcau_signature_struct_para_init(pkcau_signature_parameter_struct* init_para);
Function descriptions	initialize signature parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	signature parameter struct, the structure members can refer to Table 3-533. Structure pkcau_signature_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU signature parameter struct with a default value */
pkcau_signature_parameter_struct pkcau_signature_parameter;
pkcau_signature_struct_para_init(&pkcau_signature_parameter);
```

pkcau_hash_struct_para_init

The description of pkcau_hash_struct_para_init is shown as below:

Table 3-545. Function pkcau_hash_struct_para_init

Function name	pkcau_hash_struct_para_init
Function prototype	void pkcau_hash_struct_para_init(pkcau_hash_parameter_struct* init_para);
Function descriptions	initialize hash parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	hash parameter struct, the structure members can refer to Table 3-534. Structure pkcau_hash_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU hash parameter struct with a default value */
```

```
pkcau_hash_parameter_struct pkcau_hash_parameter;
```

```
pkcau_hash_struct_para_init(&pkcau_hash_parameter);
```

pkcau_mod_reduc_struct_para_init

The description of pkcau_mod_reduc_struct_para_init is shown as below:

Table 3-546. Function pkcau_mod_reduc_struct_para_init

Function name	pkcau_mod_reduc_struct_para_init
Function prototype	void pkcau_mod_reduc_struct_para_init(pkcau_mod_reduc_parameter_struct* init_para);
Function descriptions	initialize modular reduction parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
init_para	modular reduction parameter struct, the structure members can refer to Table 3-528. Structure pkcau_mod_reduc_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize PKCAU modular reduction parameter struct with a default value */
```

```
pkcau_mod_reduc_parameter_struct pkcau_mod_reduc_parameter;
```

```
pkcau_mod_reduc_struct_para_init(&pkcau_mod_reduc_parameter);
```

pkcau_enable

The description of pkcau_enable is shown as below:

Table 3-547. Function pkcau_enable

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

Example:

```
/* enable PKCAU */
pkcau_enable();
```

pkcau_disable

The description of pkcau_disable is shown as below:

Table 3-548. Function pkcau_disable

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

Example:

```
/* disable PKCAU */
```

```
pkcau_disable();
```

pkcau_start

The description of pkcau_start is shown as below:

Table 3-549. Function pkcau_start

Function name	pkcau_start
Function prototype	void pkcau_start(void);
Function descriptions	start operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start operation */
pkcau_start();
```

pkcau_mode_set

The description of pkcau_mode_set is shown as below:

Table 3-550. Function pkcau_mode_set

Function name	pkcau_mode_set
Function prototype	void pkcau_mode_set(uint32_t mode);
Function descriptions	configure the PKCAU operation mode
Precondition	-
The called functions	-
Input parameter{in}	
mode	PKCAU operation mode
<i>PKCAU_MODE_MOD_EXP</i>	Montgomery parameter computation then modular exponentiation
<i>PKCAU_MODE_MONT_PARAM</i>	Montgomery parameter computation only
<i>PKCAU_MODE_MOD_EXP_FAST</i>	modular exponentiation only
<i>PKCAU_MODE_CRT_EXP</i>	RSA CRT exponentiation
<i>PKCAU_MODE_MOD_</i>	modular inversion

<i>INVERSION</i>	
<i>PKCAU_MODE_ARITHMETIC_ADD</i>	arithmetic addition
<i>PKCAU_MODE_ARITHMETIC_SUB</i>	arithmetic subtraction
<i>PKCAU_MODE_ARITHMETIC_MUL</i>	arithmetic multiplication
<i>PKCAU_MODE_ARITHMETIC_COMP</i>	arithmetic comparison
<i>PKCAU_MODE_MODULAR_REDUCTION</i>	modular reduction
<i>PKCAU_MODE_MODULAR_ADD</i>	modular addition
<i>PKCAU_MODE_MODULAR_SUB</i>	modular subtraction
<i>PKCAU_MODE_MONTGOMERY_MUL</i>	Montgomery multiplication
<i>PKCAU_MODE_ECC_MUL</i>	Montgomery parameter computation then ECC scalar multiplication
<i>PKCAU_MODE_ECC_MUL_FAST</i>	ECC scalar multiplication only
<i>PKCAU_MODE_ECDSA_A_SIGN</i>	ECDSA sign
<i>PKCAU_MODE_ECDSA_A_VERIFICATION</i>	ECDSA verification
<i>PKCAU_MODE_POINT_CHECK</i>	point on elliptic curve Fp check
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PKCAU operation mode as PKCAU_MODE_ARITHMETIC_ADD */
pkcau_mode_set(PKCAU_MODE_ARITHMETIC_ADD);
```

pkcau_mont_param_operation

The description of pkcau_mont_param_operation is shown as below:

Table 3-551. Function pkcau_mont_param_operation

Function name	pkcau_mont_param_operation
Function prototype	void pkcau_mont_param_operation(pkcau_mont_parameter_struct

	*mont_para);
Function descriptions	execute montgomery parameter operation
Precondition	-
The called functions	-
Input parameter{in}	
mont_para	montgomery parameter struct, the structure members can refer to Table 3-524. Structure pkcau_mont_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the montgomery parameter structure */
pkcau_mont_parameter_struct pkcau_mont_parameter;
pkcau_mont_struct_para_init(&pkcau_mont_parameter);

/* initialize the montgomery parameters */
pkcau_mont_parameter.modulus_len = ME_MOD_SIZE;
pkcau_mont_parameter.modulus_n = (uint8_t *)modulus_n;

/* execute montgomery parameter operation */
pkcau_mont_param_operation(&pkcau_mont_parameter);

```

pkcau_mod_operation

The description of pkcau_mod_operation is shown as below:

Table 3-552. Function pkcau_mod_operation

Function name	pkcau_mod_operation
Function prototype	void pkcau_mod_operation(pkcau_mod_parameter_struct *mod_para, uint32_t mode);
Function descriptions	execute modular operation, include modular addition, modular subtraction and montgomery multiplication
Precondition	-
The called functions	-
Input parameter{in}	
mod_para	modular parameter struct, the structure members can refer to Table 3-525. Structure pkcau_mod_parameter_struct.
Input parameter{in}	
mode	modular operation mode
PKCAU_MODE_MOD_	modular addition

<i>ADD</i>	
<i>PKCAU_MODE_MOD_SUB</i>	modular subtraction
<i>PKCAU_MODE_MONT_MUL</i>	Montgomery multiplication
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the modular parameter structure */

pkcau_mod_parameter_struct pkcau_mod_parameter;

pkcau_mod_struct_para_init(&pkcau_mod_parameter);

/* initialize the modular parameters */

pkcau_mod_parameter.oprd_a = (uint8_t *)oprd_a;
pkcau_mod_parameter.oprd_b = (uint8_t *)oprd_b;
pkcau_mod_parameter.modulus_len = MA_MOD_SIZE;
pkcau_mod_parameter.modulus_n = (uint8_t *)modulus_n;

/* execute modular addition operation */

pkcau_mod_operation(&pkcau_mod_parameter, PKCAU_MODE_MOD_ADD);

```

pkcau_mod_exp_operation

The description of pkcau_mod_exp_operation is shown as below:

Table 3-553. Function pkcau_mod_exp_operation

Function name	pkcau_mod_exp_operation
Function prototype	void pkcau_mod_exp_operation(pkcau_mod_exp_parameter_struct *mod_exp_para, uint32_t mode);
Function descriptions	execute modular exponentation operation
Precondition	-
The called functions	-
Input parameter{in}	
mod_exp_para	modular exponentation parameter struct, the structure members can refer to Table 3-526. Structure pkcau_mod_exp_parameter_struct .
Input parameter{in}	
mode	modular exponentation operation mode
<i>PKCAU_MODE_MOD_</i>	montgomery parameter computation then modular exponentiation

<i>EXP</i>	
<i>PKCAU_MODE_MOD_</i> <i>EXP_FAST</i>	modular exponentiation only
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the modular exponentiation parameter structure */
pkcau_mod_exp_parameter_struct pkcau_mod_exp_parameter;
pkcau_mod_exp_struct_para_init(&pkcau_mod_exp_parameter);

/* initialize the modular exponentiation parameters */
pkcau_mod_exp_parameter.oprd_a = (uint8_t *)oprd_a;
pkcau_mod_exp_parameter.exp_e = (uint8_t *)exp_e;
pkcau_mod_exp_parameter.e_len = ME_E_SIZE;
pkcau_mod_exp_parameter.modulus_len = ME_MOD_SIZE;
pkcau_mod_exp_parameter.modulus_n = (uint8_t *)modulus_n;
pkcau_mod_exp_parameter.mont_para = (uint8_t *)mont_para;

/* execute modular exponentiation fast operation */
pkcau_mod_exp_operation(&pkcau_mod_exp_parameter,
PKCAU_MODE_MOD_EXP_FAST);

```

pkcau_mod_inver_operation

The description of pkcau_mod_inver_operation is shown as below:

Table 3-554. Function pkcau_mod_inver_operation

Function name	pkcau_mod_inver_operation
Function prototype	void void pkcau_mod_inver_operation(pkcau_mod_inver_parameter_struct *mod_inver_para);
Function descriptions	execute modular inversion operation
Precondition	-
The called functions	-
Input parameter{in}	
mod_inver_para	modular inversion parameter struct, the structure members can refer to Table 3-527. Structure pkcau_mod_inver_parameter_struct.
Output parameter{out}	
-	-

Return value	
-	-

Example:

```

/* initialize the modular inversion parameter structure */

pkcau_mod_inver_parameter_struct pkcau_mod_inver_parameter;

pkcau_mod_inver_struct_para_init(&pkcau_mod_inver_parameter);

/* initialize the modular parameters */

pkcau_inver_exp_parameter.oprd_a = (uint8_t *)oprd_a;

pkcau_inver_exp_parameter.modulus_len = ME_MOD_SIZE;

pkcau_inver_exp_parameter.modulus_n = (uint8_t *)modulus_n;

/* execute modular inversion operation */

pkcau_mod_inver_operation(&pkcau_mod_inver_parameter);

```

pkcau_mod_reduc_operation

The description of pkcau_mod_reduc_operation is shown as below:

Table 3-555. Function pkcau_mod_reduc_operation

Function name	pkcau_mod_reduc_operation
Function prototype	void pkcau_mod_reduc_operation(pkcau_mod_reduc_parameter_struct *mod_reduc_para);
Function descriptions	execute modular reduction operation
Precondition	-
The called functions	-
Input parameter{in}	
mod_reduc_para	modular reduction parameter struct, the structure members can refer to Table 3-528. Structure pkcau_mod_reduc_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the modular inversion parameter structure */

pkcau_mod_reduc_parameter_struct pkcau_mod_reduc_parameter;

pkcau_mod_reduc_struct_para_init(&pkcau_mod_reduc_parameter);

/* initialize the modular parameters */

```

```
pkcau_mod_reduc_parameter.oprd_a = (uint8_t *)oprd_a;

pkcau_mod_reduc_parameter.oprd_a_len = MA_A_SIZE;

pkcau_mod_reduc_parameter.modulus_len = MA_MOD_SIZE;

pkcau_mod_reduc_parameter.modulus_n = (uint8_t *)modulus_n;

/* execute modular reduction operation */

pkcau_mod_reduc_operation(&pkcau_mod_reduc_parameter);
```

pkcau_arithmetic_operation

The description of pkcau_arithmetic_operation is shown as below:

Table 3-556. Function pkcau_arithmetic_operation

Function name	pkcau_arithmetic_operation
Function prototype	void pkcau_arithmetic_operation(pkcau_arithmetic_parameter_struct *arithmetic_para, uint32_t mode);
Function descriptions	execute arithmetic operation
Precondition	-
The called functions	-
Input parameter{in}	
arithmetic_para	arithmetic parameter struct, the structure members can refer to Table 3-529. Structure pkcau_arithmetic_parameter_struct .
Input parameter{in}	
mode	arithmetic operation mode
<i>PKCAU_MODE_ARITHMETIC_ADD</i>	arithmetic addition
<i>PKCAU_MODE_ARITHMETIC_SUB</i>	arithmetic subtraction
<i>PKCAU_MODE_ARITHMETIC_MUL</i>	arithmetic multiplication
<i>PKCAU_MODE_ARITHMETIC_COMP</i>	arithmetic comparison
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the arithmetic parameter structure */

pkcau_arithmetic_parameter_struct pkcau_arithmetic_parameter;

pkcau_arithmetic_struct_para_init(&pkcau_arithmetic_parameter);
```

```
/* initialize the arithmetic addition parameters */

pkcau_ari_multi_parameter.oprd_a = (uint8_t *)oprd_a;

pkcau_ari_multi_parameter.oprd_b = (uint8_t *)oprd_b;

pkcau_ari_multi_parameter.oprd_len = AM_B_SIZE;

/* execute arithmetic addition operation */

pkcau_arithmetic_operation(&pkcau_arithmetic_parameter,
PKCAU_MODE_ARITHMETIC_ADD);
```

pkcau_crt_exp_operation

The description of pkcau_crt_exp_operation is shown as below:

Table 3-557. Function pkcau_crt_exp_operation

Function name	pkcau_crt_exp_operation
Function prototype	void pkcau_crt_exp_operation(pkcau_crt_parameter_struct* crt_para);
Function descriptions	execute RSA CRT exponentiation operation
Precondition	-
The called functions	-
Input parameter{in}	
crt_para	CRT parameter struct, the structure members can refer to Table 3-530. Structure pkcau_crt_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the arithmetic parameter structure */

pkcau_crt_parameter_struct pkcau_crt_parameter;

pkcau_crt_exp_operation(&pkcau_crt_parameter);

/* initialize the input ECC curve parameters */

pkcau_crt_parameter.oprd_a = (uint8_t *)rsa_crt_a;

pkcau_crt_parameter.oprd_dp = (uint8_t *)rsa_crt_dp;

pkcau_crt_parameter.oprd_dq = (uint8_t *)rsa_crt_dq;

pkcau_crt_parameter.oprd_qinv = (uint8_t *)rsa_crt_qinv;

pkcau_crt_parameter.oprd_p = (uint8_t *)rsa_crt_p;

pkcau_crt_parameter.oprd_q = (uint8_t *)rsa_crt_q;
```

```
pkcau crt_parameter.oprd_len = sizeof(rsa crt_a);
```

```
/* execute RSA CRT exponentiation operation */
```

```
pkcau crt_exp_operation(&pkcau crt_parameter);
```

pkcau_point_check_operation

The description of pkcau_point_check_operation is shown as below:

Table 3-558. Function pkcau_point_check_operation

Function name	pkcau_point_check_operation
Function prototype	void pkcau_point_check_operation(pkcau_point_parameter_struct* point_para, const pkcau_ec_group_parameter_struct* curve_group_para);
Function descriptions	execute point check operation
Precondition	-
The called functions	-
Input parameter{in}	
point_para	point parameter struct, the structure members can refer to Table 3-532. Structure pkcau_point_parameter_struct.
Input parameter{in}	
curve_group_para	ECC curve parameter struct, the structure members can refer to Table 3-531. Structure pkcau_ec_group_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize point parameter and ECC curve parameter structure */
```

```
pkcau_point_parameter_struct pkcau_point_parameter;
```

```
pkcau_ec_group_parameter_struct pkcau_curve_group;
```

```
pkcau_point_struct_para_init(&pkcau_point_parameter);
```

```
pkcau_ec_group_struct_para_init(&pkcau_curve_group);
```

```
/* initialize the input point parameter */
```

```
pkcau_point_check_parameter.point_x = pcheck_x;
```

```
pkcau_point_check_parameter.point_y = pcheck_y;
```

```
/* initialize the input ECC curve parameter */
```

```
pkcau_curve_group.modulus_p = (uint8_t *)brainpoolp256r1_p;
```

```
pkcau_curve_group.coff_a = (uint8_t *)brainpoolp256r1_a;
```

```
pkcau_curve_group.coff_b      = (uint8_t *)brainpoolp256r1_b;

pkcau_curve_group.a_sign      = 0;

pkcau_curve_group.modulus_p_len = sizeof(brainpoolp256r1_p);

/* execute point check operation */

pkcau_point_check_operation(&pkcau_point_parameter, &pkcau_curve_group);
```

pkcau_point_mul_operation

The description of pkcau_point_mul_operation is shown as below:

Table 3-559. Function pkcau_point_mul_operation

Function name	pkcau_point_mul_operation
Function prototype	void pkcau_point_mul_operation(pkcau_point_parameter_struct *point_para, const pkcau_ec_group_parameter_struct* curve_group_para, uint32_t mode);
Function descriptions	execute point multiplication operation
Precondition	-
The called functions	-
Input parameter{in}	
point_para	point parameter struct, the structure members can refer to Table 3-532. Structure pkcau_point_parameter_struct .
Input parameter{in}	
curve_group_para	ECC curve parameter struct, the structure members can refer to Table 3-531. Structure pkcau_ec_group_parameter_struct .
Input parameter{in}	
mode	point multiplication operation mode
PKCAU_MODE_ECC_MUL	montgomery parameter computation then ECC scalar multiplication
PKCAU_MODE_ECC_MUL_FAST	ECC scalar multiplication only
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize point parameter and ECC curve parameter structure */

pkcau_point_parameter_struct pkcau_point_parameter;

pkcau_ec_group_parameter_struct pkcau_curve_group;

pkcau_point_struct_para_init(&pkcau_point_parameter);
```



```
pkcau_ec_group_struct_para_init(&pkcau_curve_group);

/* initialize the input point parameter */

pkcau_point_parameter.point_x = ec_pmul_x;

pkcau_point_parameter.point_y = ec_pmul_y;

/* initialize the input ECC curve parameter */

pkcau_curve_group.modulus_p      = (uint8_t *)brainpoolp256r1_p;
pkcau_curve_group.coff_a         = (uint8_t *)brainpoolp256r1_a;
pkcau_curve_group.modulus_p_len = sizeof(brainpoolp256r1_p);
pkcau_curve_group.a_sign         = 0;
pkcau_curve_group.multi_k        = ec_pmul_k;
pkcau_curve_group.k_len          = PMUL_K_SIZE;

/* execute scalar multiplication operation */

pkcau_point_mul_operation(&pkcau_point_parameter, &pkcau_curve_group,
PKCAU_MODE_ECC_MUL);
```

pkcau_ecdsa_sign_operation

The description of pkcau_ecdsa_sign_operation is shown as below:

Table 3-560. Function pkcau_ecdsa_sign_operation

Function name	pkcau_ecdsa_sign_operation
Function prototype	void pkcau_ecdsa_sign_operation(const uint8_t* p_key_d, const uint8_t* k, pkcau_hash_parameter_struct* hash_para, const pkcau_ec_group_parameter_struct* curve_group_para);
Function descriptions	execute ECDSA sign operation
Precondition	-
The called functions	-
Input parameter{in}	
p_key_d	private key d
Input parameter{in}	
k	integer k
Input parameter{in}	
hash_para	hash parameter struct, the structure members can refer to Table 3-534. Structure pkcau_hash_parameter_struct .
Input parameter{in}	
curve_group_para	ECC curve parameter struct, the structure members can refer to Table 3-531. Structure pkcau_ec_group_parameter_struct .
Output parameter{out}	

-	-
Return value	
-	-

Example:

```

/* initialize hash parameter and ECC curve parameter structure */

pkcau_hash_parameter_struct pkcau_hash_parameter;

pkcau_ec_group_parameter_struct pkcau_curve_group;

pkcau_hash_struct_para_init(&pkcau_hash_parameter);

pkcau_ec_group_struct_para_init(&pkcau_curve_group);

/* initialize the input hash parameter */

pkcau_hash_parameter.hash_z      = hash;

pkcau_hash_parameter.hash_z_len = DATA_SIZE;

/* initialize the input ECC curve parameter */

pkcau_curve_group.modulus_p      = secp112r2_p;

pkcau_curve_group.coff_a         = secp112r2_a;

pkcau_curve_group.a_sign         = 0;

pkcau_curve_group.base_point_x   = secp112r2_gx;

pkcau_curve_group.base_point_y   = secp112r2_gy;

pkcau_curve_group.order_n        = secp112r2_n,

pkcau_curve_group.modulus_p_len  = sizeof(secp112r2_p);

pkcau_curve_group.order_n_len    = sizeof(secp112r2_n);

/* execute ECDSA sign operation */

pkcau_ecdsa_sign_operation(d, k, &pkcau_hash_parameter, &pkcau_curve_group);

```

pkcau_ecdsa_verification_operation

The description of pkcau_ecdsa_verification_operation is shown as below:

Table 3-561. Function pkcau_ecdsa_verification_operation

Function name	pkcau_ecdsa_verification_operation
Function prototype	void pkcau_ecdsa_verification_operation(pkcau_point_parameter_struct *point_para, pkcau_hash_parameter_struct *hash_para, pkcau_signature_parameter_struct *signature_para, const pkcau_ec_group_parameter_struct* curve_group_para);

Function descriptions	execute ECDSA verification operation
Precondition	-
The called functions	-
Input parameter{in}	
point_para	point parameter struct, the structure members can refer to Table 3-532. Structure pkcau_point_parameter_struct.
Input parameter{in}	
hash_para	hash parameter struct, the structure members can refer to Table 3-534. Structure pkcau_hash_parameter_struct.
Input parameter{in}	
signature_para	signature parameter struct, the structure members can refer to Table 3-533. Structure pkcau_signature_parameter_struct.
Input parameter{in}	
curve_group_para	ECC curve parameter struct, the structure members can refer to Table 3-531. Structure pkcau_ec_group_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize point parameter, hash parameter and ECC curve parameter structure */
pkcau_point_parameter_struct pkcau_point_parameter;
pkcau_hash_parameter_struct pkcau_hash_parameter;
pkcau_signature_parameter_struct pkcau_signature_parameter;
pkcau_ec_group_parameter_struct pkcau_curve_group;
pkcau_point_struct_para_init(&pkcau_point_parameter);
pkcau_hash_struct_para_init(&pkcau_hash_parameter);
pkcau_signature_struct_para_init(&pkcau_signature_parameter);
pkcau_ec_group_struct_para_init(&pkcau_curve_group);

/* initialize the input point parameters */
pkcau_point_parameter.point_x = ecc_verify_x;
pkcau_point_parameter.point_y = ecc_verify_y;

/* initialize the input hash parameters */
pkcau_hash_parameter.hash_z      = (uint8_t *)ecc_verify_hash;
pkcau_hash_parameter.hash_z_len = sizeof(ecc_verify_hash);

```

```

/* initialize the input ECC signature parameters */

pkcau_signature_parameter.sign_r = (uint8_t *)ecc_verify_r;

pkcau_signature_parameter.sign_s = (uint8_t *)ecc_verify_s;

/* initialize the input ECC curve parameters */

pkcau_curve_group.modulus_p      = (uint8_t *)brainpoolp256r1_p;
pkcau_curve_group.coff_a         = (uint8_t *)brainpoolp256r1_a;
pkcau_curve_group.a_sign         = 0;
pkcau_curve_group.base_point_x   = (uint8_t *)brainpoolp256r1_gx;
pkcau_curve_group.base_point_y   = (uint8_t *)brainpoolp256r1_gy;
pkcau_curve_group.order_n        = (uint8_t *)brainpoolp256r1_n,
pkcau_curve_group.modulus_p_len  = sizeof(brainpoolp256r1_p);
pkcau_curve_group.order_n_len    = sizeof(brainpoolp256r1_n);

/* execute ECDSA verification operation */

pkcau_ecdsa_verification_operation(&pkcau_point_parameter, &pkcau_hash_parameter,
&pkcau_signature_parameter, &pkcau_curve_group);

```

pkcau_memread

The description of pkcau_memread is shown as below:

Table 3-562. Function pkcau_memread

Function name	pkcau_memread
Function prototype	void pkcau_memread(uint32_t offset, uint8_t buf[], uint32_t size);
Function descriptions	read result from PKCAU RAM
Precondition	-
The called functions	-
Input parameter{in}	
offset	RAM address offset to PKCAU base address
Input parameter{in}	
size	number of byte to read
Output parameter{out}	
buf	big endian result buffer, the left most byte from the PKCAU should be in the first element of buffer
Return value	
-	-

Example:

```

/* read results from RAM address */

```

```
uint8_t verify_res = 0;

pkcau_memread(0x5B0, &verify_res, 1);
```

pkcau_flag_get

The description of pkcau_flag_get is shown as below:

Table 3-563. Function pkcau_flag_get

Function name	pkcau_flag_get
Function prototype	FlagStatus pkcau_flag_get(uint32_t flag);
Function descriptions	get PKCAU flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	PKCAU flags
<i>PKCAU_FLAG_ADDRE RR</i>	address error flag
<i>PKCAU_FLAG_RAME RR</i>	PKCAU RAM error flag
<i>PKCAU_FLAG_END</i>	end of PKCAU operation flag
<i>PKCAU_FLAG_BUSY</i>	busy flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get PKCAU flag status */

FlagStatus flag_state = RESET;

flag_state = pkcau_flag_get(PKCAU_FLAG_ADDRERR);
```

pkcau_flag_clear

The description of pkcau_flag_clear is shown as below:

Table 3-564. Function pkcau_flag_clear

Function name	pkcau_flag_clear
Function prototype	void pkcau_flag_clear(uint32_t flag);
Function descriptions	clear PKCAU flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	PKCAU flags

<i>PKCAU_FLAG_ADDRE RR</i>	address error flag
<i>PKCAU_FLAG_RAME RR</i>	PKCAU RAM error flag
<i>PKCAU_FLAG_END</i>	end of PKCAU operation flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear address error flag*/
pkcau_flag_clear(PKCAU_FLAG_ADDRERR);
```

pkcau_interrupt_enable

The description of pkcau_interrupt_enable is shown as below:

Table 3-565. Function pkcau_interrupt_enable

Function name	pkcau_interrupt_enable
Function prototype	void pkcau_interrupt_enable(uint32_t interrupt);
Function descriptions	enable PKCAU interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	interrupt type
<i>PKCAU_INT_ADDRER R</i>	address error interrupt
<i>PKCAU_INT_RAMERR</i>	PKCAU RAM error interrupt
<i>PKCAU_INT_END</i>	end of PKCAU operation interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable PKCAU address error interrupt */
pkcau_interrupt_enable(PKCAU_INT_ADDRERR);
```

pkcau_interrupt_disable

The description of pkcau_interrupt_disable is shown as below:

Table 3-566. Function `pkcau_interrupt_disable`

Function name	<code>pkcau_interrupt_disable</code>
Function prototype	<code>void pkcau_interrupt_disable(uint32_t interrupt);</code>
Function descriptions	disable PKCAU interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	interrupt type
<code>PKCAU_INT_ADDRER R</code>	address error interrupt
<code>PKCAU_INT_RAMERR</code>	PKCAU RAM error interrupt
<code>PKCAU_INT_END</code>	end of PKCAU operation interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable PKCAU address error interrupt */
pkcau_interrupt_disable(PKCAU_INT_ADDRERR);
```

`pkcau_interrupt_flag_get`

The description of `pkcau_interrupt_flag_get` is shown as below:

Table 3-567. Function `pkcau_interrupt_flag_get`

Function name	<code>pkcau_interrupt_flag_get</code>
Function prototype	<code>FlagStatus pkcau_interrupt_flag_get(uint32_t int_flag);</code>
Function descriptions	get PKCAU interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	PKCAU interrupt flags
<code>PKCAU_INT_FLAG_AD DRERR</code>	address error interrupt flag
<code>PKCAU_INT_FLAG_RA MERR</code>	PKCAU RAM error interrupt flag
<code>PKCAU_INT_FLAG_EN D</code>	end of PKCAU operation interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get PKCAU interrupt flag status */

FlagStatus flag_state = RESET;

flag_state = pkcau_interrupt_flag_get(PKCAU_INT_FLAG_ADDRERR);
```

pkcau_interrupt_flag_clear

The description of pkcau_interrupt_flag_clear is shown as below:

Table 3-568. Function pkcau_interrupt_flag_clear

Function name	pkcau_interrupt_flag_clear
Function prototype	void pkcau_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear PKCAU flag interrupt status
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	PKCAU interrupt flags
<i>PKCAU_INT_FLAG_ADDRERR</i>	address error interrupt flag
<i>PKCAU_INT_FLAG_RAMERR</i>	PKCAU RAM error interrupt flag
<i>PKCAU_INT_FLAG_END</i>	end of PKCAU operation interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear address error interrupt flag*/

pkcau_interrupt_flag_clear(PKCAU_INT_FLAG_ADDRERR);
```

3.19. PMU

According to the Power management unit (PMU), provides five types of power saving modes, including Sleep, Deep-sleep, SRAM_sleep, WIFI_sleep and Standby mode. The PMU registers are listed in chapter [3.19.1](#), the PMU firmware functions are introduced in chapter [3.19.2](#).

3.19.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

Table 3-569. PMU Registers

Registers	Descriptions
PMU_CTL0	Control register 0
PMU_CS0	Control and status register 0
PMU_CTL1	Control register 1
PMU_CS1	Control and status register 1
PMU_RFCTL	PMU RF control register
PMU_SECCFG	PMU secure configuration register
PMU_PRIVCFG	PMU privilege configuration register

3.19.2. Descriptions of Peripheral functions

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

Table 3-570. PMU firmware function

Function name	Function description
pmu_deinit	deinitialize the PMU
pmu_lvd_select	select low voltage detector threshold
pmu_lvd_disable	disable PMU lvd
pmu_vlvd_enable	enable PMU VLVD
pmu_vlvd_disable	disable PMU VLVD
pmu_ldo_output_select	select LDO output voltage
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_wifi_power_enable	enable WIFI power
pmu_wifi_power_disable	disable WIFI power
pmu_wifi_sram_control	WIFI & SRAM low power control
pmu_rf_force_enable	RF sequence force open/close
pmu_rf_force_disable	disable RF sequence open/close force
pmu_rf_sequence_config	RF sequence configuration
pmu_security_enable	enable the security attribution
pmu_security_disable	disable the security attribution
pmu_privilege_enable	enable the privileged access
pmu_privilege_disable	disable the privileged access
pmu_flag_reset	reset flag bit
pmu_flag_get	get flag state

pmu_deinit

The description of pmu_deinit is shown as below:

Table 3-571. 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 3-572. Function pmu_lvd_select

Function name	pmu_lvd_select
Function prototype	void pmu_lvd_select(uint32_t lvd_t_n);
Function descriptions	select low voltage detector threshold
Precondition	-
The called functions	-
Input parameter{in}	
lvd_t_n	voltage threshold value
<i>PMU_LVDT_0</i>	voltage threshold is 2.1V
<i>PMU_LVDT_1</i>	voltage threshold is 2.3V
<i>PMU_LVDT_2</i>	voltage threshold is 2.4V
<i>PMU_LVDT_3</i>	voltage threshold is 2.6V
<i>PMU_LVDT_4</i>	voltage threshold is 2.7V
<i>PMU_LVDT_5</i>	voltage threshold is 2.9V
<i>PMU_LVDT_6</i>	voltage threshold is 3.0V
<i>PMU_LVDT_7</i>	voltage threshold is 3.1V
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* select low voltage detector threshold as 3.1V */
```

```
pmu_lvd_select (PMU_LVDT_7);
```

pmu_lvd_disable

The description of pmu_lvd_disable is shown as below:

Table 3-573. 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_vlvd_enable

The description of pmu_vlvd_enable is shown as below:

Table 3-574. Function pmu_vlvd_enable

Function name	pmu_vlvd_enable
Function prototype	void pmu_vlvd_enable(void);
Function descriptions	enable PMU VLVD
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable PMU VLVD */
```

```
pmu_vlvd_enable();
```

pmu_vlvd_disable

The description of pmu_vlvd_disable is shown as below:

Table 3-575. Function pmu_vlvd_disable

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

Example:

```
/* disable PMU VLVD */
```

```
pmu_vlvd_disable();
```

pmu_ldo_output_select

The description of pmu_ldo_output_select is shown as below:

Table 3-576. Function pmu_ldo_output_select

Function name	pmu_ldo_output_select
Function prototype	void pmu_ldo_output_select(uint32_t ldo_output);
Function descriptions	select LDO output voltage
Precondition	-
The called functions	-
Input parameter{in}	
ldo_output	LDO output voltage mode
<i>PMU_LDOVS_LOW</i>	LDO output voltage low mode
<i>PMU_LDOVS_HIGH</i>	LDO output voltage high mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* LDO output low voltage */
pmu_ldo_output_select(PMU_LDOVS_LOW);
```

pmu_to_sleepmode

The description of pmu_to_sleepmode is shown as below:

Table 3-577. 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
<i>WFI_CMD</i>	use WFI command
<i>WFE_CMD</i>	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 3-578. Function pmu_to_deepsleepmode

Function name	pmu_to_deepsleepmode
Function prototype	void pmu_to_deepsleepmode(uint32_t ldo, uint32_t lowdrive, uint8_t deepsleepmodecmd);
Function descriptions	PMU work at deepsleep mode
Precondition	-
The called functions	-
Input parameter{in}	
ldo	ldo work mode
<i>PMU_LDO_NORMAL</i>	LDO normal work when pmu enter deepsleep mode
<i>PMU_LDO_LOWPOWER</i>	LDO work at low power mode when pmu enter deepsleep mode

Input parameter{in}	
lowdrive	ldo low-driver mode
<i>PMU_LOWDRIVER_DISABLE</i>	Low-driver mode disable in deep-sleep mode
<i>PMU_LOWDRIVER_ENABLE</i>	Low-driver mode enable in deep-sleep mode
Input parameter{in}	
deepsleepmodecmd	command to enter deepsleep mode
<i>WFI_CMD</i>	use WFI command
<i>WFE_CMD</i>	use WFE command
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PMU work at deepsleep mode */
```

```
pmu_to_deepsleepmode(PMU_LDO_NORMAL, PMU_LOWDRIVER_ENABLE, WFI_CMD);
```

pmu_to_standbymode

The description of pmu_to_standbymode is shown as below:

Table 3-579. 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 3-580. Function pmu_wakeup_pin_enable

Function name	pmu_wakeup_pin_enable
Function prototype	void pmu_wakeup_pin_enable(uint32_t wakeup_pin);
Function descriptions	enable wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_pin	the pin to wakeup system
<i>PMU_WAKEUP_PIN0</i>	WKUP Pin 0 (PA2)
<i>PMU_WAKEUP_PIN1</i>	WKUP Pin 1 (PA15)
<i>PMU_WAKEUP_PIN2</i>	WKUP Pin 2 (PB2)
<i>PMU_WAKEUP_PIN3</i>	WKUP Pin 3 (PA12)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable wakeup pin 0 */
pmu_wakeup_pin_enable(PMU_WAKEUP_PIN0);
```

pmu_wakeup_pin_disable

The description of pmu_wakeup_pin_disable is shown as below:

Table 3-581. Function pmu_wakeup_pin_disable

Function name	pmu_wakeup_pin_disable
Function prototype	void pmu_wakeup_pin_disable(uint32_t wakeup_pin);
Function descriptions	disable wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_pin	the pin to wakeup system
<i>PMU_WAKEUP_PIN0</i>	WKUP Pin 0 (PA2)
<i>PMU_WAKEUP_PIN1</i>	WKUP Pin 1 (PA15)
<i>PMU_WAKEUP_PIN2</i>	WKUP Pin 2 (PB2)
<i>PMU_WAKEUP_PIN3</i>	WKUP Pin 3 (PA12)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable wakeup pin 0 */
```

```
pmu_wakeup_pin_disable(PMU_WAKEUP_PIN0);
```

pmu_backup_write_enable

The description of pmu_backup_write_enable is shown as below:

Table 3-582. 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 3-583. 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_wifi_power_enable

The description of pmu_wifi_power_enable is shown as below:

Table 3-584. Function pmu_wifi_power_enable

Function name	pmu_wifi_power_enable
Function prototype	void pmu_wifi_power_enable(void);
Function descriptions	WIFI power enable
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable WIFI power */
pmu_wifi_power_enable();
```

pmu_wifi_power_disable

The description of pmu_wifi_power_disable is shown as below:

Table 3-585. Function pmu_wifi_power_disable

Function name	pmu_wifi_power_disable
Function prototype	void pmu_wifi_power_disable(void);
Function descriptions	WIFI power disable
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable WIFI power */
pmu_wifi_power_disable();
```

pmu_wifi_sram_control

The description of pmu_wifi_sram_control is shown as below:

Table 3-586. Function pmu_wifi_sram_control

Function name	pmu_wifi_sram_control
Function prototype	void pmu_wifi_sram_control(uint32_t wifi_sram);
Function descriptions	WIFI SRAM control
Precondition	-
The called functions	-
Input parameter{in}	
wifi_sram	WIFI or SRAM sleep control
<i>PMU_WIFI_SLEEP</i>	WIFI go to sleep
<i>PMU_WIFI_WAKE</i>	WIFI wakeup
<i>PMU_SRAM1_SLEEP</i>	SRAM1 go to sleep
<i>PMU_SRAM1_WAKE</i>	SRAM1 wakeup
<i>PMU_SRAM2_SLEEP</i>	SRAM2 go to sleep
<i>PMU_SRAM2_WAKE</i>	SRAM2 wakeup
<i>PMU_SRAM3_SLEEP</i>	SRAM3 go to sleep
<i>PMU_SRAM3_WAKE</i>	SRAM3 wakeup
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SRAM2 go to sleep */
```

```
pmu_wifi_sram_control(PMU_SRAM2_SLEEP);
```

pmu_rf_force_enable

The description of pmu_rf_force_enable is shown as below:

Table 3-587. Function pmu_rf_force_enable

Function name	pmu_rf_force_enable
Function prototype	void pmu_rf_force_enable(uint32_t force);
Function descriptions	enable RF sequence force open/close
Precondition	-
The called functions	-
Input parameter{in}	
force	enable RF power control
<i>PMU_RF_FORCE_OP EN</i>	Software force start, open RF power
<i>PMU_RF_FORCE_CL</i>	Software force close, close RF power

<i>OSE</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable open RF power */
```

```
pmu_rf_force_enable(PMU_RF_FORCE_OPEN);
```

pmu_rf_force_disable

The description of pmu_rf_force_disable is shown as below:

Table 3-588. Function pmu_rf_force_disable

Function name	pmu_rf_force_disable
Function prototype	void pmu_rf_force_disable(uint32_t force);
Function descriptions	disable RF sequence open/close force
Precondition	-
The called functions	-
Input parameter{in}	
force	disable RF power control
<i>PMU_RF_FORCE_OPEN</i>	Software force start, open RF power
<i>PMU_RF_FORCE_CLOSE</i>	Software force close, close RF power
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable open RF power */
```

```
pmu_rf_force_disable(PMU_RF_FORCE_OPEN);
```

pmu_rf_sequence_config

The description of pmu_rf_sequence_config is shown as below:

Table 3-589. Function pmu_rf_sequence_config

Function name	pmu_rf_sequence_config
Function prototype	void pmu_rf_sequence_config(uint32_t mode);
Function descriptions	RF sequence configuration
Precondition	-

The called functions	-
Input parameter{in}	
mode	RF sequence mode
<i>PMU_RF_SOFTWARE</i>	RF software sequence enable
<i>PMU_RF_HARDWARE</i>	RF hardware sequence enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RF hardware sequence */
```

```
pmu_rf_sequence_config(PMU_RF_HARDWARE);
```

pmu_security_enable

The description of pmu_security_enable is shown as below:

Table 3-590. Function pmu_security_enable

Function name	pmu_security_enable
Function prototype	void pmu_security_enable(uint32_t security);
Function descriptions	enable the security attribution
Precondition	-
The called functions	-
Input parameter{in}	
security	security attribution configuration
<i>PMU_SEC_WAKEUP_PIN0</i>	WKUP pin 0 security
<i>PMU_SEC_WAKEUP_PIN1</i>	WKUP pin 1 security
<i>PMU_SEC_WAKEUP_PIN2</i>	WKUP pin 2 security
<i>PMU_SEC_WAKEUP_PIN3</i>	WKUP pin 3 security
<i>PMU_SEC_LPLDO_DP_SLP_STB</i>	Low-power mode security
<i>PMU_SEC_LVD_VLVD</i>	Voltage detection and monitoring security
<i>PMU_SEC_BACKUP_WRITE</i>	Backup Domain write access security
<i>PMU_SEC_WIFI_SRAM_CONTROL</i>	WiFi_sleep and SRAM_sleep mode security
<i>PMU_SEC_RF_SEQUENCE</i>	RF security

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable WKUP pin 0 security */
```

```
pmu_security_enable(PMU_SEC_WAKEUP_PIN0);
```

pmu_security_disable

The description of pmu_security_disable is shown as below:

Table 3-591. Function pmu_security_disable

Function name	pmu_security_disable
Function prototype	void pmu_security_disable(uint32_t security);
Function descriptions	disable the security attribution
Precondition	-
The called functions	-
Input parameter{in}	
security	security attribution configuration
PMU_SEC_WAKEUP_PIN0	WKUP pin 0 security
PMU_SEC_WAKEUP_PIN1	WKUP pin 1 security
PMU_SEC_WAKEUP_PIN2	WKUP pin 2 security
PMU_SEC_WAKEUP_PIN3	WKUP pin 3 security
PMU_SEC_LPLDO_DP_SLP_STB	Low-power mode security
PMU_SEC_LVD_VLVD	Voltage detection and monitoring security
PMU_SEC_BACKUP_WRITE	Backup Domain write access security
PMU_SEC_WIFI_SRAM_CONTROL	WIFI_sleep and SRAM_sleep mode security
PMU_SEC_RF_SEQUENCE	RF security
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable WKUP pin 0 security */
```

```
pmu_security_disable(PMU_SEC_WAKEUP_PIN0);
```

pmu_privilege_enable

The description of pmu_privilege_enable is shown as below:

Table 3-592. Function pmu_privilege_enable

Function name	pmu_privilege_enable
Function prototype	void pmu_privilege_enable(void);
Function descriptions	enable the privileged access only
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the privileged access only */
```

```
pmu_privilege_enable();
```

pmu_privilege_disable

The description of pmu_privilege_disable is shown as below:

Table 3-593. Function pmu_privilege_disable

Function name	pmu_privilege_disable
Function prototype	void pmu_privilege_disable(void);
Function descriptions	disable the privileged access only
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the privileged access only */
```

```
pmu_privilege_disable();
```

pmu_flag_reset

The description of pmu_flag_reset is shown as below:

Table 3-594. Function pmu_flag_reset

Function name	pmu_flag_reset
Function prototype	void pmu_flag_reset(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_reset(PMU_FLAG_RESET_WAKEUP);
```

pmu_flag_get

The description of pmu_flag_get is shown as below:

Table 3-595. Function pmu_flag_get

Function name	pmu_flag_get
Function prototype	FlagStatus pmu_flag_get(uint32_t pmu_flag);
Function descriptions	get flag state
Precondition	-
The called functions	-
Input parameter{in}	
pmu_flag	flag
PMU_FLAG_WAKEUP	wakeup flag
PMU_FLAG_STANDBY	standby flag
PMU_FLAG_LVD	low voltage detector status flag
PMU_FLAG_VLVD	VDDA low voltage flag
PMU_FLAG_LDOVSRF	LDO voltage select ready flag

<i>PMU_FLAG_LDRF</i>	low-driver mode ready flag
<i>PMU_FLAG_WIFI_SLEEP</i>	WIFI is in sleep state
<i>PMU_FLAG_WIFI_ACTIVE</i>	WIFI is in active state
<i>PMU_FLAG_SRAM1_SLEEP</i>	SRAM1 is in sleep state
<i>PMU_FLAG_SRAM1_ACTIVE</i>	SRAM1 is in active state
<i>PMU_FLAG_SRAM2_SLEEP</i>	SRAM2 is in sleep state
<i>PMU_FLAG_SRAM2_ACTIVE</i>	SRAM2 is in active state
<i>PMU_FLAG_SRAM3_SLEEP</i>	SRAM3 is in sleep state
<i>PMU_FLAG_SRAM3_ACTIVE</i>	SRAM3 is in active state
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get flag state */
```

```
FlagStatus status;
```

```
status = pmu_flag_get(PMU_FLAG_WAKEUP);
```


3.20. QSPI

The QSPI is a specialized interface that communicate with Flash memories. This interface support single, dual or quad SPI FLASH. The QSPI registers are listed in chapter [3.20.1](#), the QSPI firmware functions are introduced in chapter [3.20.2](#).

3.20.1. Descriptions of Peripheral registers

QSPI registers are listed in the table shown as below:

Table 3-596. QSPI registers

Registers	Descriptions
QSPI_CTL	QSPI control register
QSPI_DCFG	QSPI device configuration register
QSPI_STAT	QSPI status register
QSPI_STATC	QSPI status clear register
QSPI_DTLEN	QSPI data length register
QSPI_TCFG	QSPI transfer configuration register
QSPI_ADDR	QSPI address register
QSPI_ALTE	QSPI alternate bytes register
QSPI_DATA	QSPI data register
QSPI_STATMK	QSPI status mask register
QSPI_STATMATCH	QSPI status match register
QSPI_INTERVAL	QSPI interval register
QSPI_TMOUT	QSPI timeout register
QSPI_FLUSH	QSPI fifo flush register
QSPI_WTCNT	QSPI wait cnt for indirect wire mode register
QSPI_SPTMOUT	QSPI timeout for status polling mode register
QSPI_FMC_SECCFG	QSPI FMC mode security configuration register
QSPI_CTLF	QSPI control register in FMC mode
QSPI_TCFGF	QSPI transfer configuration register in FMC mode
QSPI_ALTEF	QSPI alternate bytes register in FMC mode
QSPI_BYTE_CNT	QSPI complete bytes counter register
QSPI_PRIVCFG	QSPI privilege configuration register
QSPI_STAT_SEC	QSPI secure status register in FMC mode
QSPI_STATC_SEC	QSPI secure status clear register in FMC mode
QSPI_DTLEN_SEC	QSPI secure data length register in FMC mode
QSPI_TCFG_SEC	QSPI secure transfer configuration register in FMC mode
QSPI_ADDR_SEC	QSPI secure address register in FMC mode
QSPI_ALTE_SEC	QSPI secure alternate bytes register in FMC mode
QSPI_DATA_SEC	QSPI secure data register in FMC mode

3.20.2. Descriptions of Peripheral functions

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

Table 3-597. QSPI firmware function

Function name	Function description
qspi_deinit	reset QSPI peripheral
qspi_init_struct_para_init	initialize the parameters of QSPI init struct with the default values
qspi_init	initialize QSPI peripheral parameter
qspi_enable	enable QSPI
qspi_disable	disable QSPI
qspi_dma_enable	enable QSPI DMA function
qspi_dma_disable	disable QSPI DMA function
qspi_command	QSPI command parameter configure
qspi_transmit	QSPI transmit data
qspi_receive	QSPI receive data
qspi_autopolling	configure QSPI autopolling mode
qspi_memorymapped	configure QSPI memorymapped mode
qspi_abort	abort the current transmission
qspi_command_fmc_s	QSPI command parameter configure in FMC mode
qspi_transmit_fmc_s	QSPI transmit data in FMC mode
qspi_receive_fmc_s	QSPI receive data in FMC mode
qspi_autopolling_fmc_s	configure QSPI autopolling mode in FMC mode
qspi_memorymapped_fmc_s	configure QSPI memorymapped mode in FMC mode
qspi_interrupt_enable	enable QSPI interrupt
qspi_interrupt_disable	disable QSPI interrupt
qspi_flag_get	get QSPI flag status
qspi_flag_clear	clear QSPI flag status

Structure qspi_init_struct

Table 3-598. qspi_init_struct

Member name	Function description
prescaler	QSPI prescaler (0x00 - 0xFF)
fifothreshold	QSPI FIFO threshold (0x01 - 0x1F)
sampleshift	QSPI sample shift (QSPI_SAMPLE_SHIFTING_NONE, QSPI_SAMPLE_SHIFTING_HALFCYCLE, QSPI_SAMPLE_SHIFTING_ONECYCLE)
flashsize	external flash size

Member name	Function description
	(0x00 - 0x1F)
chipselecthightime	CLK cycles which the chip select (nCS) must stay high between two command sequences (QSPI_CS_HIGH_TIME_n_CYCLE (n=1,2,...,8))
clockmode	QSPI clock mode (QSPI_CLOCK_MODE_0, QSPI_CLOCK_MODE_3)

Structure qspi_command_struct

Table 3-599. qspi_command_struct

Member name	Function description
instructionmode	instruction mode (QSPI_INSTRUCTION_NONE, QSPI_INSTRUCTION_1_LINE, QSPI_INSTRUCTION_2_LINES, QSPI_INSTRUCTION_4_LINES)
instruction	8 bits instruction (0x00-0xFF)
addressmode	address mode (QSPI_ADDRESS_NONE, QSPI_ADDRESS_1_LINE, QSPI_ADDRESS_2_LINES, QSPI_ADDRESS_4_LINES)
addresssize	address size (QSPI_ADDRESS_8_BITS, QSPI_ADDRESS_16_BITS, QSPI_ADDRESS_24_BITS, QSPI_ADDRESS_32_BITS)
address	address to be send to the external Flash memory
alternatebytemode	alternate bytes mode (QSPI_ALTERNATE_BYTES_NONE, QSPI_ALTERNATE_BYTES_1_LINE, QSPI_ALTERNATE_BYTES_2_LINES, QSPI_ALTERNATE_BYTES_4_LINES)
addresssize	alternate bytes size (QSPI_ADDRESS_8_BITS, QSPI_ADDRESS_16_BITS, QSPI_ADDRESS_24_BITS, QSPI_ADDRESS_32_BITS)
alternatebytes	alternate bytes information
dummyscycles	dummy cycles (0x00 - 0x1F)
datamode	data mode (QSPI_DATA_NONE, QSPI_DATA_1_LINE, QSPI_DATA_2_LINES, QSPI_DATA_4_LINES)
nbdata	32 bits data length
sioomode	Send instruction only once mode (QSPI_SIOO_INST_EVERY_CMD, QSPI_SIOO_INST_ONLY_FIRST_CMD)

Structure `qspi_autopolling_struct`

Table 3-600. `qspi_autopolling_struct`

Member name	Function description
match	match value (0x0-0xFFFFFFFF)
mask	mask value (0x0-0xFFFFFFFF)
interval	number of clock cycles between two read during automatic polling phases (0x0-0xFFFF)
stautsbytessize	the size of the status bytes received (0x01-0x04)
matchmode	method used for determining a match (QSPI_MATCH_MODE_AND, QSPI_MATCH_MODE_OR)
automaticstop	if automatic polling is stopped after a match (QSPI_AUTOMATIC_STOP_DISABLE, QSPI_AUTOMATIC_STOP_ENABLE)

`qspi_deinit`

The description of `qspi_deinit` is shown as below:

Table 3-601. Function `qspi_deinit`

Function name	<code>qspi_deinit</code>
Function prototype	<code>void qspi_deinit(void);</code>
Function descriptions	reset QSPI peripheral
Precondition	-
The called functions	<code>rcu_periph_reset_enable</code> / <code>rcu_periph_reset_disable</code>
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset QSPI */
qspi_deinit();
```

`qspi_init_struct_para_init`

The description of `qspi_init_struct_para_init` is shown as below:

Table 3-602. Function `qspi_init_struct_para_init`

Function name	<code>qspi_init_struct_para_init</code>
---------------	---

Function prototype	void qspi_init_struct_para_init(qspi_init_struct* qspi_struct);
Function descriptions	Initialize the parameters of QSPI init struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
qspi_struct	QSPI init parameter struct, the structure members can refer to Table 3-598. qspi_init_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the parameters of QSPI */

qspi_parameter_struct qspi_init_struct;

qspi_struct_para_init(&qspi_init_struct);

```

qspi_init

The description of qspi_init is shown as below:

Table 3-603. Function qspi_init

Function name	qspi_init
Function prototype	void qspi_init(qspi_init_struct* qspi_struct);
Function descriptions	Initialize QSPI peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
qspi_struct	QSPI parameter initialization struct, the structure members can refer to members of the structure Table 3-598. qspi_init_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize QSPI */

qspi_init_struct qspi_struct;

qspi_struct.clockmode = QSPI_CLOCK_MODE_0;

qspi_struct.prescaler = 3;

qspi_struct.fifothreshold = 16;

```

```

qspi_struct.sampleshift = QSPI_SAMPLE_SHIFTING_NONE;

qspi_struct.flashsize = 0x1F;

qspi_struct.chipselecthightime = QSPI_CS_HIGH_TIME_1_CYCLE;

qspi_init(&qspi_struct);

```

qspi_enable

The description of qspi_enable is shown as below:

Table 3-604. Function quad_spi_enable

Function name	qspi_enable
Function prototype	void qspi_enable(void);
Function descriptions	Enable QSPI
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable QSPI */

qspi_enable();

```

qspi_disable

The description of qspi_disable is shown as below:

Table 3-605. Function qspi_disable

Function name	qspi_disable
Function prototype	void qspi_disable(void);
Function descriptions	Disable QSPI
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable QSPI */

qspi_disable();
```

qspi_dma_enable

The description of qspi_dma_enable is shown as below:

Table 3-606. Function qspi_dma_enable

Function name	qspi_dma_enable
Function prototype	void qspi_dma_enable(void);
Function descriptions	Enable QSPI DMA function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable QSPI DMA function */

qspi_dma_enable();
```

qspi_dma_disable

The description of qspi_dma_disable is shown as below:

Table 3-607. Function qspi_dma_disable

Function name	qspi_dma_disable
Function prototype	void qspi_dma_disable(void);
Function descriptions	Disable QSPI DMA function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable QSPI DMA function */
```

```
qspi_dma_disable();
```

qspi_command

The description of qspi_command is shown as below:

Table 3-608. Function qspi_command

Function name	qspi_command
Function prototype	void qspi_command(qspi_command_struct* command_struct);
Function descriptions	send QSPI command
Precondition	-
The called functions	-
Input parameter{in}	
command_struct	QSPI command struct, the structure members can refer to Table 3-599. qspi_command_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* send QSPI command */
```

```
qspi_command_struct command;
```

```
command.instruction_mode = QSPI_INSTRUCTION_1_LINE;
```

```
command.instruction = RDID;
```

```
command.addressmode = QSPI_ADDRESS_NONE;
```

```
command.addr_size = QSPI_ADDRESS_SIZE_8_BITS;
```

```
command.address = 0;
```

```
command.alternatebytemode = QSPI_ALTERNATE_BYTES_1_LINE;
```

```
command.alternatebytessize = QSPI_ALTERNATE_BYTES_24_BITS;
```

```
command.alter = 0;
```

```
command.dummy_cycles = 0;
```

```
command.data_mode = QSPI_DATA_1_LINE;
```

```
command.data_length = 3;
```

```
command.Sioomode = QSPI_SIOO_INST_EVERY_CMD;
```

```
qspi_command(&command);
```


qspi_transmit

The description of qspi_transmit is shown as below:

Table 3-609. Function qspi_transmit

Function name	qspi_transmit
Function prototype	void qspi_transmit(uint8_t *tdata);
Function descriptions	QSPI transmit data
Precondition	-
The called functions	-
Input parameter{in}	
tdata	transmit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* QSPI transmit data */
qspi_transmit(&tdata);
```

qspi_receive

The description of qspi_receive is shown as below:

Table 3-610. Function qspi_receive

Function name	qspi_receive
Function prototype	void qspi_receive(uint8_t *rdata)
Function descriptions	QSPI receive data
Precondition	-
The called functions	-
Input parameter{in}	
rdata	receive data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* QSPI receive data */
qspi_receive(&rdata);
```

qspi_autopolling

The description of qspi_autopolling is shown as below:

Table 3-611. Function qspi_command

Function name	qspi_autopolling
Function prototype	void qspi_autopolling(qspi_command_struct *cmd, qspi_autopolling_struct *cfg);
Function descriptions	configure QSPI autopolling mode
Precondition	-
The called functions	-
Input parameter{in}	
cmd	QSPI command struct, the structure members can refer to Table 3-599. qspi_command_struct
Input parameter{in}	
cfg	QSPI command struct, the structure members can refer to Table 3-600. qspi_autopolling_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* send QSPI command */

qspi_command_struct command;

qspi_autopolling_struct autopolling_cmd;

command.instruction_mode = QSPI_INSTRUCTION_1_LINE;

command.instruction = RDID;

command.addressmode = QSPI_ADDRESS_NONE;

command.addr_size = QSPI_ADDRESS_SIZE_8_BITS;

command.address = 0;

command.alternatebytemode = QSPI_ALTERNATE_BYTES_1_LINE;

command.alternatebytessize = QSPI_ALTERNATE_BYTES_24_BITS;

command.alter = 0;

command.dummy_cycles = 0;

command.data_mode = QSPI_DATA_1_LINE;

command.data_length = 3;

```

```

command. Siomode = QSPI_SIOO_INST_EVERY_CMD;

autopolling_cmd.match          = 0x02;

autopolling_cmd.mask           = 0x02;

autopolling_cmd.matchmode      = QSPI_MATCH_MODE_AND;

autopolling_cmd.statusbytesize = 1;

autopolling_cmd.interval       = 0x10;

autopolling_cmd.automaticstop   = QSPI_AUTOMATIC_STOP_ENABLE;

qspi_autopolling(&command, &autopolling_cmd);

```

qspi_memorymapped

The description of qspi_memorymapped is shown as below:

Table 3-612. Function qspi_memorymapped

Function name	qspi_memorymapped
Function prototype	void qspi_memorymapped(qspi_command_struct *cmd, uint16_t timeout, uint32_t toen);
Function descriptions	configure QSPI memorymapped mode
Precondition	-
The called functions	-
Input parameter{in}	
cmd	QSPI command struct, the structure members can refer to Table 3-599. qspi_command_struct
Input parameter{in}	
timeout	QSPI command timeout value (0-0xFFFF)
Input parameter{in}	
toen	Timeout counter enable
QSPI_TOEN_DISABLE	Timeout counter disable
QSPI_TOEN_ENABLE	Timeout counter enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* config QSPI memorymapped */

qspi_command_struct command;

command.instruction_mode = QSPI_INSTRUCTION_1_LINE;

```

```

command.instruction = RDID;

command.addressmode = QSPI_ADDRESS_NONE;

command.addr_size = QSPI_ADDRESS_SIZE_8_BITS;

command.address = 0;

command.alternatebytemode = QSPI_ALTERNATE_BYTES_1_LINE;

command.alternatebytessize = QSPI_ALTERNATE_BYTES_24_BITS;

command.alter = 0;

command.dummy_cycles = 0;

command.data_mode = QSPI_DATA_1_LINE;

command.data_length = 3;

command.Sioomode = QSPI_SIOO_INST_EVERY_CMD;

qspi_memorymapped (&command,0xff, QSPI_TOEN_ENABLE);

```

qspi_abort

The description of qspi_abort is shown as below:

Table 3-613. Function qspi_abort

Function name	qspi_abort
Function prototype	void qspi_abort(void);
Function descriptions	abort the current transmission
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* abort QSPI */

qspi_abort();

```

qspi_command_fmc_s

The description of qspi_command_fmc_s is shown as below:

Table 3-614. Function qspi_command_fmc_s

Function name	qspi_command_fmc_s
Function prototype	void qspi_command_fmc_s (qspi_command_struct* command_struct);
Function descriptions	config QSPI command in FMC mode
Precondition	-
The called functions	-
Input parameter{in}	
command_struct	QSPI command struct, the structure members can refer to Table 3-599. qspi_command_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* config QSPI command */

qspi_command_struct command;

command.instruction_mode = QSPI_INSTRUCTION_1_LINE;

command.instruction = RDID;

command.addressmode = QSPI_ADDRESS_NONE;

command.addr_size = QSPI_ADDRESS_SIZE_8_BITS;

command.address = 0;

command.alternatebytemode = QSPI_ALTERNATE_BYTES_1_LINE;

command.alternatebytessize = QSPI_ALTERNATE_BYTES_24_BITS;

command.alter = 0;

command.dummy_cycles = 0;

command.data_mode = QSPI_DATA_1_LINE;

command.data_length = 3;

command.Sioomode = QSPI_SIOO_INST_EVERY_CMD;

qspi_command_fmc_s (&command);

```

qspi_transmit_fmc_s

The description of qspi_transmit_fmc_s is shown as below:

Table 3-615. Function qspi_transmit_fmc_s

Function name	qspi_transmit_fmc_s
----------------------	---------------------

Function prototype	void qspi_transmit_fmc_s (uint8_t *tdata);
Function descriptions	QSPI transmit data in FMC mode
Precondition	-
The called functions	-
Input parameter{in}	
tdata	transmit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* QSPI transmit data */
qspi_transmit_fmc_s(&tdata);
```

qspi_receive_fmc_s

The description of qspi_receive_fmc_s is shown as below:

Table 3-616. Function qspi_data_receive_fmc_s

Function name	qspi_receive_fmc_s
Function prototype	void qspi_receive_fmc_s (uint8_t *rdata)
Function descriptions	QSPI receive data in FMC mode
Precondition	-
The called functions	-
Input parameter{in}	
rdata	receive data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* QSPI receive data */
qspi_receive_fmc_s(&rdata);
```

qspi_autopolling_fmc_s

The description of qspi_autopolling_fmc_s is shown as below:

Table 3-617. Function qspi_command_fmc_s

Function name	qspi_autopolling_fmc_s
Function prototype	void qspi_autopolling_fmc_s (qspi_command_struct *cmd, qspi_autopolling_struct *cfg);

Function descriptions	configure QSPI autopolling mode in FMC mode
Precondition	-
The called functions	-
Input parameter{in}	
cmd	QSPI command struct, the structure members can refer to Table 3-599. qspi_command_struct
Input parameter{in}	
cfg	QSPI command struct, the structure members can refer to Table 3-600. qspi_autopolling_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* send QSPI command */

qspi_command_struct command;

qspi_autopolling_struct autopolling_cmd;

command.instruction_mode = QSPI_INSTRUCTION_1_LINE;

command.instruction = RDID;

command.addressmode = QSPI_ADDRESS_NONE;

command.addr_size = QSPI_ADDRESS_SIZE_8_BITS;

command.address = 0;

command.alternatebytemode = QSPI_ALTERNATE_BYTES_1_LINE;

command.alternatebytessize = QSPI_ALTERNATE_BYTES_24_BITS;

command.alter = 0;

command.dummy_cycles = 0;

command.data_mode = QSPI_DATA_1_LINE;

command.data_length = 3;

command.Sioomode = QSPI_SIOO_INST_EVERY_CMD;

autopolling_cmd.match          = 0x02;

autopolling_cmd.mask           = 0x02;

autopolling_cmd.matchmode      = QSPI_MATCH_MODE_AND;

autopolling_cmd.statusbytessize = 1;

```

```

autopolling_cmd.interval      = 0x10;

autopolling_cmd.automaticstop  = QSPI_AUTOMATIC_STOP_ENABLE;

qspi_autopolling_fmc_s (&command, &autopolling_cmd);

```

qspi_memorymapped_fmc_s

The description of qspi_memorymapped_fmc_s is shown as below:

Table 3-618. Function qspi_command_fmc_s

Function name	qspi_memorymapped_fmc_s
Function prototype	void qspi_memorymapped_fmc_s(qspi_command_struct *cmd, uint16_t timeout, uint32_t toen);
Function descriptions	configure QSPI memorymapped mode in FMC mode
Precondition	-
The called functions	-
Input parameter{in}	
cmd	QSPI command struct, the structure members can refer to Table 3-599. qspi_command_struct
Input parameter{in}	
timeout	QSPI command timeout value (0-0xFFFF)
Input parameter{in}	
toen	Timeout counter enable
<i>QSPI_TOEN_DISABLE</i>	Timeout counter disable
<i>QSPI_TOEN_ENABLE</i>	Timeout counter enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* send QSPI memorymapped */

qspi_command_struct command;

command.instruction_mode = QSPI_INSTRUCTION_1_LINE;

command.instruction = RDID;

command.addressmode = QSPI_ADDRESS_NONE;

command.addr_size = QSPI_ADDRESS_SIZE_8_BITS;

command.address = 0;

command.alternatebytemode = QSPI_ALTERNATE_BYTES_1_LINE;

```



```
command.alternatebytesize = QSPI_ALTERNATE_BYTES_24_BITS;

command.alter = 0;

command.dummy_cycles = 0;

command.data_mode = QSPI_DATA_1_LINE;

command.data_length = 3;

command.sioomode = QSPI_SIOO_INST_EVERY_CMD;

qspi_memorymapped_fmc_s(&command, 0xff, QSPI_TOEN_ENABLE);
```

qspi_interrupt_enable

The description of qspi_interrupt_enable is shown as below:

Table 3-619. Function qspi_interrupt_enable

Function name	qspi_interrupt_enable
Function prototype	void qspi_interrupt_enable(uint8_t interrupt);
Function descriptions	enable QSPI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	QSPI interrupt
QSPI_INT_TC	transfer complete interrupt
QSPI_INT_FT	FIFO threshold interrupt
QSPI_INT_TERR	transfer error interrupt
QSPI_INT_SM	status match interrupt
QSPI_INT_TMOUT	timeout interrupt
QSPI_INT_WS	wrong start sequence interrupt
Output parameter{out}	
-	-
Return value	

Example:

```
/* enable QSPI transfer complete interrupt */

qspi_interrupt_enable (QSPI_INT_TC);
```

qspi_interrupt_disable

The description of qspi_interrupt_disable is shown as below:

Table 3-620. Function qspi_interrupt_disable

Function name	qspi_interrupt_disable
---------------	------------------------

Function prototype	void qspi_interrupt_disable(uint8_t interrupt);
Function descriptions	disable QSPI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	QSPI interrupt
QSPI_INT_TC	transfer complete interrupt
QSPI_INT_FT	FIFO threshold interrupt
QSPI_INT_TERR	transfer error interrupt
QSPI_INT_SM	status match interrupt
QSPI_INT_TMOUT	timeout interrupt
QSPI_INT_WS	wrong start sequence interrupt
Output parameter{out}	
-	-
Return value	

Example:

```
/* disable QSPI transfer complete interrupt */
```

```
qspi_interrupt_disable (QSPI_INT_TC);
```

qspi_flag_get

The description of qspi_flag_get is shown as below:

Table 3-621. Function qspi_flag_get

Function name	qspi_flag_get
Function prototype	FlagStatus qspi_flag_get(uint32_t flag);
Function descriptions	get QSPI flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	QSPI flag status
QSPI_FLAG_TERR	transfer error flag
QSPI_FLAG_TC	transfer complete flag
QSPI_FLAG_FT	FIFO threshold flag
QSPI_FLAG_SM	status match flag
QSPI_FLAG_TMOUT	timeout flag
QSPI_FLAG_WS	wrong start sequence flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get QSPI transfer complete flag */  
  
while(RESET == qspi_flag_get (QSPI_FLAG_TC));
```

qspi_flag_clear

The description of qspi_flag_clear is shown as below:

Table 3-622. Function qspi_flag_clear

Function name	qspi_flag_clear
Function prototype	void qspi_flag_clear(uint32_t flag);
Function descriptions	Clear QSPI flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	QSPI flag status
QSPI_FLAG_TERR	transfer error flag
QSPI_FLAG_TC	transfer complete flag
QSPI_FLAG_SM	status match flag
QSPI_FLAG_TMOUT	timeout flag
QSPI_FLAG_WS	wrong start sequence flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear QSPI transfer complete flag status */  
  
qspi_flag_clear(QSPI_FLAG_TC);
```

3.21. 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 [3.21.1](#), the RCU firmware functions are introduced in chapter [3.21.2](#).

3.21.1. Descriptions of Peripheral registers

Table 3-623. RCU Registers

Registers	Descriptions
RCU_CTL	control register

Registers	Descriptions
RCU_PLL	PLL register
RCU_CFG0	clock configuration register 0
RCU_INT	clock interrupt register
RCU_AHB1RST	AHB1 reset register
RCU_AHB2RST	AHB2 reset register
RCU_AHB3RST	AHB3 reset register
RCU_APB1RST	APB1 reset register
RCU_APB2RST	APB2 reset register
RCU_AHB1EN	AHB1 enable register
RCU_AHB2EN	AHB2 enable register
RCU_AHB3EN	AHB3 enable register
RCU_APB1EN	APB1 enable register
RCU_APB2EN	APB2 enable register
RCU_AHB1SPEN	AHB1 sleep mode enable register
RCU_AHB2SPEN	AHB2 sleep mode enable register
RCU_AHB3SPEN	AHB3 sleep mode enable register
RCU_APB1SPEN	APB1 sleep mode enable register
RCU_APB2SPEN	APB2 sleep mode enable register
RCU_BDCTL	backup domain control register
RCU_RSTSCK	reset source/clock register
RCU_PLLSSCTL	PLL clock spread spectrum control register
RCU_PLLCFG	PLL clock configuration register
RCU_CFG1	clock configuration register 1
RCU_ADDCTL	additional clock control register
RCU_SECP_CFG	secure configuration register
RCU_SECP_STAT	secure status register
RCU_AHB1SECP_STAT	AHB1 secure status register
RCU_AHB2SECP_STAT	AHB2 secure status register
RCU_AHB3SECP_STAT	AHB3 secure status register
RCU_APB1SECP_STAT	APB1 secure status register
RCU_APB2SECP_STAT	APB2 secure status register
RCU_VKEY	voltage key register
RCU_DSV	deep-sleep mode voltage register

3.21.2. Descriptions of Peripheral functions

Table 3-624. 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

Function name	Function description
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_hxtal_plli2s_enable	enable HXTAL for PLLI2S
rcu_hxtal_plli2s_disable	disable HXTAL for PLLI2S
rcu_hxtal_pllp_enable	enable HXTAL for system CK_PLLP
rcu_hxtal_pllp_disable	disable HXTAL for system CK_PLLP
rcu_control_unit_powerup	power on the clock
rcu_control_unit_powerdown	power down the clock
rcu_rfpll_cal_enable	enable the RF PLL calculation
rcu_rfpll_cal_disable	disable the RF PLL calculation
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_ckout1_config	configure the CK_OUT1 clock source and divider
rcu_pll_config	configure the main PLL clock
rcu_plli2s_config	configure the PLLI2S clock
rcu_plldig_config	configure the PLLDIG clock
rcu_plldig_div_sys_config	configure PLLDIG clock divider factor for system clock
rcu_rtc_clock_config	configure the RTC clock source selection
rcu_rtc_div_config	configure the frequency division of RTC clock when HXTAL was selected as its clock source
rcu_i2s_clock_config	configure the I2S clock source selection
rcu_pll_div_i2s_config	configure the PLL divider factor for I2S clock
rcu_hpdf_clock_config	configure the HPDF clock source selection (not support on GD32W515Tx series devices)
rcu_hpdf_audio_clock_config	configure the HPDF AUDIO clock source selection(not support on GD32W515Tx series devices)
rcu_sdio_clock_config	configure the SDIO clock source selection
rcu_sdio_div_config	configure the frequency division of the sdio source clock
rcu_usbfs_clock_config	configure the usb clock source selection
rcu_usbfs_div_config	configure the frequency division of the usbfs source clock
rcu_i2c0_clock_config	configure the I2C0 clock source selection
rcu_usart0_clock_config	configure the USART0 clock source selection
rcu_usart2_clock_config	configure the USART2 clock source selection

Function name	Function description
rcu_irc16m_div_config	configure IRC16M clock divider factor for system clock
rcu_timer_clock_prescaler_config	configure the TIMER clock prescaler selection
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_irc16m_adjust_value_set	set the IRC16M adjust value
rcu_spread_spectrum_config	configure the spread spectrum modulation for the main PLL clock
rcu_spread_spectrum_enable	enable the spread spectrum modulation for the main PLL clock
rcu_spread_spectrum_disable	disable the spread spectrum modulation for the main PLL clock
rcu_hxtal_clock_monitor_enable	enable the HXTAL clock monitor
rcu_hxtal_clock_monitor_disable	disable the HXTAL clock monitor
rcu_rf_hxtal_clock_monitor_enable	enable the RF HXTAL clock monitor
rcu_rf_hxtal_clock_monitor_disable	disable the RF HXTAL clock monitor
rcu_voltage_key_unlock	unlock the voltage key
rcu_deepsleep_voltage_set	set the deep sleep mode voltage
rcu_clock_freq_get	get the system clock, bus clock frequency
rcu_flag_get	get the clock stabilization and peripheral reset flags
rcu_security_enable	enable the security attribution
rcu_security_disable	disable the security attribution
rcu_privilege_enable	enable the privileged access
rcu_privilege_disable	disable the privileged access
rcu_all_reset_flag_clear	clear all the reset flag
rcu_interrupt_flag_get	get the clock stabilization interrupt and ckm flags
rcu_interrupt_flag_clear	clear the interrupt flags
rcu_security_flag_get	get the secure status flag
rcu_interrupt_enable	enable the stabilization interrupt
rcu_interrupt_disable	disable the stabilization interrupt

Enum rcu_periph_enum

Table 3-625. Enum rcu_periph_enum

enum name	Function description
RCU_GPIOA	GPIOA clock
RCU_GPIOB	GPIOB clock
RCU_GPIOC	GPIOC clock

enum name	Function description
RCU_TZPCU	TZPCU clock
RCU_TSI	TSI clock
RCU_CRC	CRC clock
RCU_WIFI	WIFI clock
RCU_WIFIRUN	WIFIRUN clock
RCU_SRAM0	SRAM0 clock
RCU_SRAM1	SRAM1 clock
RCU_SRAM2	SRAM2 clock
RCU_SRAM3	SRAM3 clock
RCU_DMA0	DMA0 clock
RCU_DMA1	DMA1 clock
RCU_USBFS	USBFS clock
RCU_DCI	DCI clock(DCI not support on GD32W515Tx series devices)
RCU_PKCAU	PKCAU clock
RCU_CAU	CAU clock
RCU_HAU	HAU clock
RCU_TRNG	TRNG clock
RCU_SQPI	SQPI clock
RCU_QSPI	QSPI clock
RCU_TIMER1	TIMER1 clock
RCU_TIMER2	TIMER2 clock
RCU_TIMER3	TIMER3 clock(TIMER3 not support on GD32W515Tx series devices)
RCU_TIMER4	TIMER4 clock
RCU_TIMER5	TIMER5 clock
RCU_WWDGT	WWDGT clock
RCU_SPI1	SPI1 clock
RCU_USART1	USART1 clock
RCU_USART0	USART0 clock
RCU_I2C0	I2C0 clock
RCU_I2C1	I2C1 clock
RCU_PMU	PMU clock
RCU_RTC	RTC clock
RCU_TIMER0	TIMER0 clock
RCU_USART2	USART0 clock
RCU_ADC	ADC clock
RCU_SDIO	SDIO clock
RCU_SPI0	SPI0 clock
RCU_SYSCFG	SYSCFG clock
RCU_TIMER15	TIMER15 clock
RCU_TIMER16	TIMER16 clock
RCU_HPDI	HPDI clock(HPDI not support on GD32W515Tx series devices)

enum name	Function description
RCU_RF	RF clock

Enum rcu_periph_sleep_enum

Table 3-626. Enum rcu_periph_sleep_enum

enum name	Function description
RCU_GPIOA_SLP	GPIOA clock when sleep mode
RCU_GPIOB_SLP	GPIOB clock when sleep mode
RCU_GPIOC_SLP	GPIOC clock when sleep mode
RCU_TZPCU_SLP	TZPCU clock when sleep mode
RCU_TSI_SLP	TSI clock when sleep mode
RCU_CRC_SLP	CRC clock when sleep mode
RCU_WIFI_SLP	WIFI clock when sleep mode
RCU_WIFIRUN_SLP	WIFIRUN clock when sleep mode
RCU_SRAM0_SLP	SRAM0 clock when sleep mode
RCU_SRAM1_SLP	SRAM1 clock when sleep mode
RCU_SRAM2_SLP	SRAM2 clock when sleep mode
RCU_SRAM3_SLP	SRAM3 clock when sleep mode
RCU_DMA0_SLP	DMA0 clock when sleep mode
RCU_DMA1_SLP	DMA1 clock when sleep mode
RCU_USBFS_SLP	USBFS clock when sleep mode
RCU_DCI_SLP	DCI clock when sleep mode(DCI not support on GD32W515Tx series devices)
RCU_PKCAU_SLP	PKCAU clock when sleep mode
RCU_CAU_SLP	CAU clock when sleep mode
RCU_HAU_SLP	HAU clock when sleep mode
RCU_TRNG_SLP	TRNG clock when sleep mode
RCU_SQPI_SLP	SQPI clock when sleep mode
RCU_QSPI_SLP	QSPI clock when sleep mode
RCU_TIMER1_SLP	TIMER1 clock when sleep mode
RCU_TIMER2_SLP	TIMER2 clock when sleep mode
RCU_TIMER3_SLP	TIMER3 clock when sleep mode(TIMER3 not support on GD32W515Tx series devices)
RCU_TIMER4_SLP	TIMER4 clock when sleep mode
RCU_TIMER5_SLP	TIMER5 clock when sleep mode
RCU_WWDGT_SLP	WWDGT clock when sleep mode
RCU_SPI1_SLP	SPI1 clock when sleep mode
RCU_USART1_SLP	USART1 clock when sleep mode
RCU_USART0_SLP	USART0 clock when sleep mode
RCU_I2C0_SLP	I2C0 clock when sleep mode
RCU_I2C1_SLP	I2C1 clock when sleep mode
RCU_PMU_SLP	PMU clock when sleep mode
RCU_RTC_SLP	RTC clock when sleep mode

enum name	Function description
RCU_TIMER0_SLP	TIMER0 clock when sleep mode
RCU_USART2_SLP	USART0 clock when sleep mode
RCU_ADC_SLP	ADC clock when sleep mode
RCU_SDIO_SLP	SDIO clock when sleep mode
RCU_SPI0_SLP	SPI0 clock when sleep mode
RCU_SYSCFG_SLP	SYSCFG clock when sleep mode
RCU_TIMER15_SLP	TIMER15 clock when sleep mode
RCU_TIMER16_SLP	TIMER16 clock when sleep mode
RCU_HPDP_SLP	HPDP clock when sleep mode(HPDP not support on GD32W515Tx series devices)
RCU_RF_SLP	RF clock when sleep mode

Enum rcu_periph_reset_enum

Table 3-627. Enum rcu_periph_reset_enum

enum name	Function description
RCU_GPIOARST	GPIOA clock reset
RCU_GPIOBRST	GPIOB clock reset
RCU_GPIOCRST	GPIOC clock reset
RCU_TZPCURST	TZPCU clock reset
RCU_TSIRST	TSI clock reset
RCU_CRCRST	CRC clock reset
RCU_WIFIRST	WIFI clock reset
RCU_DMA0RST	DMA0 clock reset
RCU_DMA1RST	DMA1 clock reset
RCU_USBFIRST	USBFS clock reset
RCU_DCIRST	DCI clock reset(DCI not support on GD32W515Tx series devices)
RCU_PKCAURST	PKCAU clock reset
RCU_CAURST	CAU clock reset
RCU_HAURST	HAU clock reset
RCU_TRNGRST	TRNG clock reset
RCU_SQPIRST	SQPI clock reset
RCU_QSPIRST	QSPI clock reset
RCU_TIMER1RST	TIMER1 clock reset
RCU_TIMER2RST	TIMER2 clock reset
RCU_TIMER3RST	TIMER3 clock reset(TIMER3 not support on GD32W515Tx series devices)
RCU_TIMER4RST	TIMER4 clock reset
RCU_TIMER5RST	TIMER5 clock reset
RCU_WWDGTRST	WWDGT clock reset
RCU_SPI1RST	SPI1 clock reset
RCU_USART1RST	USART1 clock reset
RCU_USART0RST	USART0 clock reset

enum name	Function description
RCU_I2C0RST	I2C0 clock reset
RCU_I2C1RST	I2C1 clock reset
RCU_PMURST	PMU clock reset
RCU_RTCRST	RTC clock reset
RCU_TIMER0RST	TIMER0 clock reset
RCU_USART2RST	USART0 clock reset
RCU_ADCRST	ADC clock reset
RCU_SDIORST	SDIO clock reset
RCU_SPI0RST	SPI0 clock reset
RCU_SYSCFGRST	SYSCFG clock reset
RCU_TIMER15RST	TIMER15 clock reset
RCU_TIMER16RST	TIMER16 clock reset
RCU_HPDRST	HPDF clock reset(HPDF not support on GD32W515Tx series devices)
RCU_RFRST	RF clock reset

Enum rcu_flag_enum

Table 3-628. Enum rcu_flag_enum

enum name	Function description
RCU_FLAG_IRC16MS TB	IRC16M stabilization flags
RCU_FLAG_HXTALST B	HXTAL stabilization flags
RCU_FLAG_PLLDIGST B	PLLDIG stabilization flags
RCU_FLAG_PLLSTB	PLL stabilization flags
RCU_FLAG_PLLI2SST B	PLLI2S stabilization flags
RCU_FLAG_LXTALST B	LXTAL stabilization flags
RCU_FLAG_IRC32KST B	IRC32K stabilization flags
RCU_FLAG_OBLRST	IRC48M stabilization flags
RCU_FLAG_EPRST	external PIN reset flags
RCU_FLAG_PORRST	power reset flags
RCU_FLAG_SWRST	software reset flags
RCU_FLAG_FWDGTR ST	FWDGT reset flags
RCU_FLAG_WWDGTR ST	WWDGT reset flags
RCU_FLAG_LPRST	low-power reset flags

Enum rcu_int_flag_enum

Table 3-629. Enum rcu_int_flag_enum

enum name	Function description
RCU_INT_FLAG_IRC3 2KSTB	IRC32K stabilization interrupt flag
RCU_INT_FLAG_LXTA LSTB	LXTAL stabilization interrupt flag
RCU_INT_FLAG_IRC1 6MSTB	IRC16M stabilization interrupt flag
RCU_INT_FLAG_HXTA LSTB	HXTAL stabilization interrupt flag
RCU_INT_FLAG_PLLS TB	PLL stabilization interrupt flag
RCU_INT_FLAG_PLLI2 SSTB	PLLI2S stabilization interrupt flag
RCU_INT_FLAG_PLLD IGSTB	PLLDIG stabilization interrupt flag
RCU_INT_FLAG_CKM	HXTAL clock stuck interrupt flag

Enum rcu_int_flag_clear_enum

Table 3-630. Enum rcu_int_flag_clear_enum

enum name	Function description
RCU_INT_FLAG_IRC3 2KSTB_CLR	IRC32K stabilization interrupt flags clear
RCU_INT_FLAG_LXTA LSTB_CLR	LXTAL stabilization interrupt flags clear
RCU_INT_FLAG_IRC1 6MSTB_CLR	IRC16M stabilization interrupt flags clear
RCU_INT_FLAG_HXTA LSTB_CLR	HXTAL stabilization interrupt flags clear
RCU_INT_FLAG_PLLS TB_CLR	PLL stabilization interrupt flags clear
RCU_INT_FLAG_PLLI2 SSTB_CLR	PLLI2S stabilization interrupt flags clear
RCU_INT_FLAG_PLLD IGSTB_CLR	PLLDIG stabilization interrupt flags clear
RCU_INT_FLAG_CKM _CLR	CKM interrupt flags clear

Enum rcu_int_enum

Table 3-631. Enum rcu_int_enum

enum name	Function description
RCU_INT_IRC32KSTB	IRC32K stabilization interrupt
RCU_INT_LXTALSTB	LXTAL stabilization interrupt
RCU_INT_IRC16MSTB	IRC16M stabilization interrupt
RCU_INT_HXTALSTB	HXTAL stabilization interrupt
RCU_INT_PLLSTB	PLL stabilization interrupt
RCU_INT_PLLI2SSTB	PLLI2S stabilization interrupt
RCU_INT_PLLDIGSTB	PLLDIG stabilization interrupt

Enum rcu_osci_type_enum

Table 3-632. Enum rcu_osci_type_enum

enum name	Function description
RCU_HXTAL	HXTAL
RCU_LXTAL	LXTAL
RCU_IRC16M	IRC16M
RCU_IRC32K	IRC32K
RCU_PLLDIG_CK	PLLDIG
RCU_PLL_CK	PLL
RCU_PLLI2S_CK	PLLI2S

Enum rcu_clock_freq_enum

Table 3-633. Enum rcu_clock_freq_enum

enum name	Function description
CK_SYS	system clock
CK_AHB	AHB clock
CK_APB1	APB1 clock
CK_APB2	APB2 clock
CK_USART0	USART0 clock
CK_USART2	USART2 clock
CK_I2C0	I2C0 clock

Enum rcu_sec_enum

Table 3-634. Enum rcu_sec_enum

enum name	Function description
RCU_SEC_IRC16MSEC	IRC16M clock configuration and status bits security
RCU_SEC_HXTALSEC	HXTAL clock configuration and status bits security
RCU_SEC_IRC32KSEC	IRC32K clock configuration and status bits security

enum name	Function description
C	
RCU_SEC_LXTALSEC	LXTAL clock configuration and status bits security
RCU_SEC_SYSCLOCKSEC	SYSCLK clock selection, STOPWUCK bit, clock output on MCO configuration
RCU_SEC_PRESCSEC	AHBx/APBx prescaler configuration bits security
RCU_SEC_PLLSEC	main PLL configuration and status bits security
RCU_SEC_PLLDIGSEC	PLLDIG configuration and status bits security
RCU_SEC_PLLI2SSEC	PLLI2S configuration and status bits security
RCU_SEC_RMVRSTFSEC	remove reset flag security
RCU_SEC_BKPSEC	BKP security

Enum rcu_sec_flag_enum

Table 3-635. Enum rcu_sec_flag_enum

enum name	Function description
RCU_SEC_FLAG_IRC16MSEC	IRC16M clock configuration and status bits security
RCU_SEC_FLAG_HXTALSEC	HXTAL clock configuration and status bits security
RCU_SEC_FLAG_IRC32KSEC	IRC32K clock configuration and status bits security
RCU_SEC_FLAG_LXTALSEC	LXTAL clock configuration and status bits security
RCU_SEC_FLAG_SYSCLOCKSEC	SYSCLK clock selection, STOPWUCK bit, clock output on MCO configuration
RCU_SEC_FLAG_PRESCSEC	AHBx/APBx prescaler configuration bits security
RCU_SEC_FLAG_PLLSEC	main PLL configuration and status bits security
RCU_SEC_FLAG_PLLDIGSEC	PLLDIG configuration and status bits security
RCU_SEC_FLAG_PLLI2SSEC	PLLI2S configuration and status bits security
RCU_SEC_FLAG_RMVRSTFSEC	remove reset flag security
RCU_SEC_FLAG_BKPSEC	BKP security
RCU_SEC_FLAG_GPIOA	GPIOA clock

enum name	Function description
RCU_SEC_FLAG_GPIOB	GPIOB clock
RCU_SEC_FLAG_GPIOC	GPIOC clock
RCU_SEC_FLAG_CRC	CRC clock
RCU_SEC_FLAG_WIFI	WIFI clock
RCU_SEC_FLAG_FMC	FMC clock
RCU_SEC_FLAG_SRAM0	SRAM0 clock
RCU_SEC_FLAG_SRAM1	SRAM1 clock
RCU_SEC_FLAG_SRAM2	SRAM2 clock
RCU_SEC_FLAG_SRAM3	SRAM3 clock
RCU_SEC_FLAG_DMA0	DMA0 clock
RCU_SEC_FLAG_DMA1	DMA1 clock
RCU_SEC_FLAG_USBFS	USBFS clock
RCU_SEC_FLAG_DCI	DCI clock(DCI not support on GD32W515Tx series devices)
RCU_SEC_FLAG_PKCAU	PKCAU clock
RCU_SEC_FLAG_CAU	CAU clock
RCU_SEC_FLAG_HAU	HAU clock
RCU_SEC_FLAG_TRNG	TRNG clock
RCU_SEC_FLAG_SQPI	SQPI clock
RCU_SEC_FLAG_QSPI	QSPI clock
RCU_SEC_FLAG_TIMER1	TIMER1 clock
RCU_SEC_FLAG_TIMER2	TIMER2 clock
RCU_SEC_FLAG_TIMER3	TIMER3 clock
RCU_SEC_FLAG_TIMER4	TIMER4 clock
RCU_SEC_FLAG_TIMER5	TIMER5 clock

enum name	Function description
RCU_SEC_FLAG_WWDGT	WWDGT clock
RCU_SEC_FLAG_SPI1	SPI1 clock
RCU_SEC_FLAG_USART1	USART1 clock
RCU_SEC_FLAG_USART0	USART0 clock
RCU_SEC_FLAG_I2C0	I2C0 clock
RCU_SEC_FLAG_I2C1	I2C1 clock
RCU_SEC_FLAG_PMU	PMU clock
RCU_SEC_FLAG_TIMER0	TIMER0 clock
RCU_SEC_FLAG_USART2	USART2 clock
RCU_SEC_FLAG_ADC	ADC clock
RCU_SEC_FLAG_SDIO	SDIO clock
RCU_SEC_FLAG_SPI0	SPI0 clock
RCU_SEC_FLAG_SYSCFG	SYSCFG clock
RCU_SEC_FLAG_TIMER15	TIMER15 clock
RCU_SEC_FLAG_TIMER16	TIMER16 clock
RCU_SEC_FLAG_HPDA	HPDA clock(HPDA not support on GD32W515Tx series devices)
RCU_SEC_FLAG_RF	RF clock

Enum rcu_unit_enum

Table 3-636. Enum rcu_unit_enum

enum name	Function description
RCU_UNIT_HXTAL	HXTAL
RCU_UNIT_PLLDIG	PLLDIG
RCU_UNIT_RFPLL	RFPLL
RCU_UNIT_LDOANA	LDOANA
RCU_UNIT_LDOCLK	LDOCLK
RCU_UNIT_BANDGAP	BANDGAP
RCU_UNIT_HXTAL	HXTAL

rcu_deinit

The description of rcu_deinit is shown as below:

Table 3-637. 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_oscstap_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 3-638. 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 3-625. Enum 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 3-639. 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 3-625. Enum 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 3-640. 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 3-626. Enum 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 3-641. 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 3-626. Enum 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 3-642. 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 3-627. Enum 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 3-643. 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 3-627. Enum 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 3-644. 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 3-645. 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_hxtal_plli2s_enable

The description of rcu_hxtal_plli2s_enable is shown as below:

Table 3-646. Function rcu_hxtal_plli2s_enable

Function name	rcu_hxtal_plli2s_enable
Function prototype	void rcu_hxtal_plli2s_enable(void);
Function descriptions	enable HXTAL for PLLI2S
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable HXTAL for PLLI2S */
```

```
rcu_hxtal_plli2s_enable();
```

rcu_hxtal_plli2s_disable

The description of rcu_hxtal_plli2s_disable is shown as below:

Table 3-647. Function rcu_hxtal_plli2s_disable

Function name	rcu_hxtal_plli2s_disable
Function prototype	void rcu_hxtal_plli2s_disable(void);
Function descriptions	disable HXTAL for PLLI2S
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable HXTAL for PLLI2S */
```

```
rcu_hxtal_plli2s_disable();
```

rcu_hxtal_pllp_enable

The description of rcu_hxtal_pllp_enable is shown as below:

Table 3-648. Function rcu_hxtal_pllp_enable

Function name	rcu_hxtal_pllp_enable
Function prototype	void rcu_hxtal_pllp_enable(void);
Function descriptions	enable HXTAL for system CK_PLLP
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable HXTAL for system CK_PLLP */
```

```
rcu_hxtal_pllp_enable();
```

rcu_hxtal_pllp_disable

The description of rcu_hxtal_pllp_disable is shown as below:

Table 3-649. Function rcu_hxtal_pllp_disable

Function name	rcu_hxtal_pllp_disable
Function prototype	void rcu_hxtal_pllp_disable(void);
Function descriptions	disable HXTAL for system CK_PLLP
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* disable HXTAL for system CK_PLLP */
```

```
rcu_hxtal_pll_p_disable();
```

rcu_control_unit_powerup

The description of rcu_control_unit_powerup is shown as below:

Table 3-650. Function rcu_control_unit_powerup

Function name	rcu_control_unit_powerup
Function prototype	void rcu_control_unit_powerup (rcu_unit_enum rcu_unit);
Function descriptions	power on the clock
Precondition	-
The called functions	-
Input parameter{in}	
rcu_unit	
RCU_UNIT_HXTAL	power on the HXTAL
RCU_UNIT_PLLDIG	power on the PLLDIG
RCU_UNIT_RFPLL	enable the RF PLL calculation
RCU_UNIT_LDOANA	power on LDO analog
RCU_UNIT_LDOCLK	power on the LDO clock
RCU_UNIT_BANDGAP	power on the BandGap
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* power on the HXTAL */
```

```
rcu_control_unit_powerup(RCU_UNIT_HXTAL);
```

rcu_control_unit_powerdown

The description of rcu_control_unit_powerdown is shown as below:

Table 3-651. Function rcu_control_unit_powerdown

Function name	rcu_control_unit_powerdown
Function prototype	void rcu_control_unit_powerdown (rcu_unit_enum rcu_unit);
Function descriptions	power down the clock
Precondition	-
The called functions	-

Input parameter{in}	
rcu_unit	
<i>RCU_UNIT_HXTAL</i>	power down the HXTAL
<i>RCU_UNIT_PLLDIG</i>	power down the PLLDIG
<i>RCU_UNIT_RFPLL</i>	disenable the RF PLL calculation
<i>RCU_UNIT_LDOANA</i>	power down LDO analog
<i>RCU_UNIT_LDOCLK</i>	power down the LDO clock
<i>RCU_UNIT_BANDGAP</i>	power down the BandGap
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* power on the HXTAL */
rcu_control_unit_powerdown(RCU_UNIT_HXTAL);
```

rcu_rfpll_cal_enable

The description of rcu_rfpll_cal_enable is shown as below:

Table 3-652. Function rcu_rfpll_cal_enable

Function name	rcu_rfpll_cal_enable
Function prototype	void rcu_rfpll_cal_enable(void);
Function descriptions	enable the RF PLL calculation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the RF PLL calculation */
rcu_rfpll_cal_enable ();
```

rcu_rfpll_cal_disable

The description of rcu_rfpll_cal_disable is shown as below:

Table 3-653. Function rcu_rfpll_cal_disable

Function name	rcu_rfpll_cal_disable
----------------------	-----------------------

Function prototype	void rcu_rfppl_cal_disable(void);
Function descriptions	disable the RF PLL calculation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the RF PLL calculation */
rcu_rfppl_cal_disable();
```

rcu_system_clock_source_config

The description of rcu_system_clock_source_config is shown as below:

Table 3-654. 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
<i>RCU_CKSYSSRC_IRC</i> <i>16M</i>	select CK_IRC16M as the CK_SYS source
<i>RCU_CKSYSSRC_HXTAL</i> <i>TAL</i>	select CK_HXTAL as the CK_SYS source
<i>RCU_CKSYSSRC_PLL</i>	select CK_PLL as the CK_SYS source
<i>RCU_CKSYSSRC_PLL</i> <i>DIG</i>	select CK_PLLDIG 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 3-655. 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_IRC16M/RCU_SCSS_HXTAL/RCU_SCSS_PLL/ RCU_CKSYSRC_PLLDIG

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 3-656. 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 3-657. 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
<i>RCU_APB1_CKAHB_D</i> <i>IV4</i>	select CK_AHB/4 as CK_APB1
<i>RCU_APB1_CKAHB_D</i> <i>IV8</i>	select CK_AHB/8 as CK_APB1
<i>RCU_APB1_CKAHB_D</i> <i>IV16</i>	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 3-658. 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
<i>RCU_APB2_CKAHB_D</i> <i>IV1</i>	select CK_AHB as CK_APB2
<i>RCU_APB2_CKAHB_D</i>	select CK_AHB/2 as CK_APB2

<i>IV2</i>	
<i>RCU_APB2_CK_AHB_D</i> <i>IV4</i>	select CK_AHB/4 as CK_APB2
<i>RCU_APB2_CK_AHB_D</i> <i>IV8</i>	select CK_AHB/8 as CK_APB2
<i>RCU_APB2_CK_AHB_D</i> <i>IV16</i>	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_CK_AHB_DIV8);
```

rcu_ckout0_config

The description of rcu_ckout0_config is shown as below:

Table 3-659. 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
<i>RCU_CKOUT0SRC_IR</i> <i>C16M</i>	select IRC16M
<i>RCU_CKOUT0SRC_LX</i> <i>TAL</i>	select LXTAL
<i>RCU_CKOUT0SRC_H</i> <i>XTAL</i>	select HXTAL
<i>RCU_CKOUT0SRC_PL</i> <i>LP</i>	select PLLP
Input parameter{in}	
ckout0_div	CK_OUT0 divider
<i>RCU_CKOUT0_DIVx</i>	CK_OUT0 is divided by x(x=1,2,3,4,5)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the HXTAL as CK_OUT0 clock source */

rcu_ckout0_config(RCU_CKOUT0SRC_HXTAL, RCU_CKOUT0_DIV1);
```

rcu_ckout1_config

The description of rcu_ckout1_config is shown as below:

Table 3-660. Function rcu_ckout1_config

Function name	rcu_ckout1_config
Function prototype	void rcu_ckout1_config(uint32_t ckout1_src);
Function descriptions	configure the CK_OUT1 clock source
Precondition	-
The called functions	-
Input parameter{in}	
ckout1_src	CK_OUT1 clock source selection
<i>RCU_CKOUT1SRC_CKSYS</i>	select system clock
<i>RCU_CKOUT1SRC_PLLI2S</i>	select PLLI2S
<i>RCU_CKOUT1SRC_HXTAL</i>	select HXTAL
<i>RCU_CKOUT1SRC_PLLDIG</i>	select PLLDIG
Input parameter{in}	
ckout1_div	CK_OUT1 divider
<i>RCU_CKOUT1_DIVx</i>	CK_OUT1 is divided by x(x=1,2,3,4,5)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the HXTAL as CK_OUT1 clock source */

rcu_ckout1_config(RCU_CKOUT1SRC_HXTAL, RCU_CKOUT1_DIV1);
```

rcu_pll_config

The description of rcu_pll_config is shown as below:

Table 3-661. Function rcu_pll_config

Function name	rcu_pll_config
Function prototype	ErrStatus rcu_pll_config(uint32_t pll_src, uint32_t pll_psc, uint32_t pll_n,

	uint32_t pll_p);
Function descriptions	configure the main PLL clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_src	PLL clock source selection
<i>RCU_PLLSRC_IRC16M</i>	IRC16M clock is selected as source clock of PLL
<i>RCU_PLLSRC_HXTAL</i>	HXTAL is selected as source clock of PLL
Input parameter{in}	
pll_psc	the PLL VCO source clock prescaler
<i>uint32_t</i>	2~63
Input parameter{in}	
pll_n	the PLL VCO clock multi factor
<i>uint32_t</i>	64~511
Input parameter{in}	
pll_p	the PLLP output frequency division factor from PLL VCO clock
<i>uint32_t</i>	2,4,6,8
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* Configure the main PLL, PSC = 8, PLL_N = 240, PLL_P = 2 */
```

```
rcu_pll_config(RCU_PLLSRC_HXTAL,8,240,2);
```

rcu_plli2s_config

The description of rcu_plli2s_config is shown as below:

Table 3-662. Function rcu_plli2s_config

Function name	rcu_plli2s_config
Function prototype	ErrStatus rcu_plli2s_config(uint32_t plli2s_n, uint32_t plli2s_psc,uint32_t plli2s_div);
Function descriptions	configure the PLLI2S clock
Precondition	-
The called functions	-
Input parameter{in}	
plli2s_n	PLLI2S clock VCO clock multiplication factor
<i>uint32_t</i>	8~127
Input parameter{in}	
plli2s_psc	PLLI2S VCO source clock pre-scale

<i>RCU_PLLI2SSRC_DIV</i> x	PLLI2S VCO source clock is divided x(x=1,2,...,8)
Input parameter{in}	
plli2s_div	PLLI2SDIV clock divider factor
<i>RCU_PLLI2S_DIVx</i>	PLLI2SDIV clock is divided x(x=1.5,2,2.5,3,...,32)
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* configure the I2S PLL, PLLI2S_PSC = 5, PLL_N = 10, PLLI2S_DIV = 2 */
rcu_plli2s_config (10, RCU_PLLI2SSRC_DIV5, RCU_PLLI2S_DIV2);
```

rcu_plldig_config

The description of rcu_plldig_config is shown as below:

Table 3-663. Function rcu_pllpresel_config

Function name	rcu_plldig_config
Function prototype	void rcu_plldig_config(uint32_t plldig_clk)
Function descriptions	configure the PLLDIG output clock frequency
Precondition	-
The called functions	-
Input parameter{in}	
plldig_clk	the PLLDIG VCO clock this parameter sselection
<i>RCU_PLLDIG_192M</i>	selected 192Mhz as PLLDIG output frequency
<i>RCU_PLLDIG_240M</i>	selected 240Mhz as PLLDIG output frequency
<i>RCU_PLLDIG_320M</i>	selected 320Mhz as PLLDIG output frequency
<i>RCU_PLLDIG_480M</i>	selected 480Mhz as PLLDIG output frequency
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* configure the PLLDIG output 192Mhz clock frequency */
rcu_plldig_config(RCU_PLLDIG_192M);
```

rcu_plldig_div_sys_config

The description of rcu_plldig_div_sys_config is shown as below:

Table 3-664. Function rcu_plldig_div_sys_config

Function name	rcu_plldig_div_sys_config
Function prototype	void rcu_plldig_div_sys_config(uint32_t plldig_div_sys);
Function descriptions	configure PLLDIG clock divider factor for system clock
Precondition	-
The called functions	-
Input parameter{in}	
plldig_div_sys	PLLDIG clock divider factor for system clock
<i>RCU_PLLDIG_SYS_DIVx</i>	PLLDIG clock divided by x for system clock(x=1,2,3,...,64)
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* configure PLLDIG clock divider 2 for system clock */
```

```
rcu_plldig_div_sys_config (RCU_PLLDIG_SYS_DIV2);
```

rcu_rtc_clock_config

The description of rcu_rtc_clock_config is shown as below:

Table 3-665. 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
<i>RCU_RTC_SRC_NONE</i>	no clock selected
<i>RCU_RTC_SRC_LXTAL</i>	CK_LXTAL selected as RTC source clock
<i>RCU_RTC_SRC_IRC32K</i>	CK_IRC32K selected as RTC source clock
<i>RCU_RTC_SRC_HXTAL_DIV_RTCDIV</i>	CK_HXTAL/RTCDIV selected as RTC source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select IRC32K as RTC clock source */
```

```
rcu_rtc_clock_config(RCU_RTCSRC_IRC32K);
```

rcu_rtc_div_config

The description of rcu_rtc_div_config is shown as below:

Table 3-666. Function rcu_rtc_div_config

Function name	rcu_rtc_div_config
Function prototype	void rcu_rtc_div_config(uint32_t rtc_div);
Function descriptions	configure the frequency division of RTC clock when HXTAL was selected as its clock source
Precondition	-
The called functions	-
Input parameter{in}	
rtc_div	RTC clock frequency division
<i>RCU_RTC_HXTAL_NONE</i>	no clock for RTC
<i>RCU_RTC_HXTAL_DIVx</i>	RTCDIV clock select CK_HXTAL/x, x = 2....31
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* RTCDIV clock select CK_HXTAL/2 */
```

```
rcu_rtc_div_config(RCU_RTC_HXTAL_DIV2);
```

rcu_i2s_clock_config

The description of rcu_i2s_clock_config is shown as below:

Table 3-667. Function rcu_i2s_clock_config

Function name	rcu_i2s_clock_config
Function prototype	void rcu_i2s_clock_config(uint32_t i2s_clock_source);
Function descriptions	configure the I2S clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
i2s_clock_source	I2S clock source selection
<i>RCU_I2SSRC_PLLI2S</i>	CK_PLLI2S selected as I2S source clock
<i>RCU_I2SSRC_I2S_CKIN</i>	external i2s_ckin pin selected as I2S source clock

<i>RCU_I2SSRC_I2S_PL</i> <i>LDIV</i>	PLL division selected as I2S source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select CK_PLLI2S as I2S source clock */
rcu_i2s_clock_config(RCU_I2SSRC_PLLI2S);
```

rcu_pll_div_i2s_config

The description of rcu_pll_div_i2s_config is shown as below:

Table 3-668. Function rcu_pll_div_i2s_config

Function name	rcu_pll_div_i2s_config
Function prototype	void rcu_pll_div_i2s_config(uint32_t pll_div_i2s);
Function descriptions	configure the PLL divider factor for I2S clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_div_i2s	PLL divider factor for I2S clock
<i>RCU_PLLDIV_I2S_DIV</i> <i>x</i>	PLL clock divided by x for I2S clock (x=1,2,...,64)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PLL clock divided by 8 for I2S clock */
rcu_pll_div_i2s_config (RCU_PLLDIV_I2S_DIV8);
```

rcu_hpdf_clock_config

The description of rcu_hpdf_clock_config is shown as below:

Table 3-669. Function rcu_hpdf_clock_config

Function name	rcu_hpdf_clock_config
Function prototype	void rcu_hpdf_clock_config(uint32_t hpdf_clock_source);
Function descriptions	configure the hpdf clock source selection
Precondition	-
The called functions	-

Input parameter{in}	
hpdf_clock_source	hpdf clock source selection
<i>RCU_HPDFSRC_PCLK2</i>	PCLK2 clock selected as HPDF source clock
<i>RCU_HPDFSRC_CKSYS</i>	system clock selected as HPDF source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select system clock as hpdf clock source */
```

```
rcu_hpdf_clock_config(RCU_HPDFSRC_CKSYS);
```

rcu_hpdf_audio_clock_config

The description of rcu_hpdf_audio_clock_config is shown as below:

Table 3-670. Function rcu_hpdf_audio_clock_config

Function name	rcu_hpdf_audio_clock_config
Function prototype	void rcu_hpdf_clock_config(uint32_t hpdf_clock_source);
Function descriptions	configure the HPDF AUDIO clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
hpdfaudio_clock_source	HPDF AUDIO clock source selection
<i>RCU_HPDAUDIOSRC_PLLI2S</i>	PLLI2S output clock selected as HPDF AUDIO source clock
<i>RCU_HPDAUDIOSRC_I2S_CKIN</i>	external I2S_CKIN PIN selected as HPDF AUDIO source clock
<i>RCU_HPDAUDIOSRC_PLL</i>	PLL division selected as HPDF AUDIO source clock
<i>RCU_HPDAUDIOSRC_IRC16M</i>	IRC16M selected as HPDF AUDIO source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select PLLI2S clock as hpdf audio clock source */
```

```
rcu_hpdf_audio_clock_config(RCU_HPDAUDIOSRC_PLLI2S);
```

rcu_sdio_clock_config

The description of rcu_sdio_clock_config is shown as below:

Table 3-671. Function rcu_sdio_clock_config

Function name	rcu_sdio_clock_config
Function prototype	void rcu_sdio_clock_config(uint32_t sdio_clock_source);
Function descriptions	SDIO clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
sdio_clock_source	SDIO clock source selection
<i>RCU_SDIOSRC_PLL</i>	PLL output clock selected as SDIO source clock
<i>RCU_SDIOSRC_PLLDIG</i>	PLLDIG selected as SDIO source clock
<i>RCU_SDIOSRC_IRC16M</i>	IRC16M selected as SDIO source clock
<i>RCU_SDIOSRC_HXTAL</i>	HXTAL selected as SDIO source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select PLL clock as sdio clock source */
rcu_sdio_clock_config(RCU_SDIOSRC_PLL);
```

rcu_sdio_div_config

The description of rcu_sdio_div_config is shown as below:

Table 3-672. Function rcu_sdio_div_config

Function name	rcu_sdio_div_config
Function prototype	void rcu_sdio_div_config(uint32_t sdio_div);
Function descriptions	configure the frequency division of the sdio source clock
Precondition	-
The called functions	-
Input parameter{in}	
sdio_div	SDIO clock frequency division
<i>RCU_SDIODIV_DIVx</i>	SDIODIV input source clock divided by x,(x = 1....32)
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* SDIODIV input source clock divided by 4 */
```

```
rcu_sdio_div_config(RCU_SDIODIV_DIV4);
```

rcu_usbfs_clock_config

The description of rcu_usbfs_clock_config is shown as below:

Table 3-673. Function rcu_usbfs_clock_config

Function name	rcu_usbfs_clock_config
Function prototype	void rcu_usbfs_clock_config(uint32_t usbfs_clock_source);
Function descriptions	configure the USBFS clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usbfs_clock_source	USBFS clock source selection
<i>RCU_USBFSRC_PLL</i>	PLL clock selected as USB source clock
<i>RCU_USBFSRC_PLL DIG</i>	PLLDIG clock selected as USB source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PLL clock selected as USB source clock */
```

```
rcu_usbfs_clock_config(RCU_USBFSRC_PLL);
```

rcu_usbfs_div_config

The description of rcu_usbfs_div_config is shown as below:

Table 3-674. Function rcu_usbfs_div_config

Function name	rcu_usbfs_div_config
Function prototype	void rcu_usbfs_div_config(uint32_t usbfs_div);
Function descriptions	configure the frequency division of the usbfs source clock
Precondition	-
The called functions	-
Input parameter{in}	
usbfs_div	USBFS clock frequency division

<i>RCU_USBFS_DIVx</i>	USBFS DIV input source clock divided by x, (x = 1,...,32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USBFS DIV input source clock divided by 4 */
```

```
rcu_usbfs_div_config(RCU_USBFS_DIV4);
```

rcu_i2c0_clock_config

The description of rcu_i2c0_clock_config is shown as below:

Table 3-675. Function rcu_i2c0_clock_config

Function name	rcu_i2c0_clock_config
Function prototype	void rcu_i2c0_clock_config(uint32_t i2c0_clock_source);
Function descriptions	configure the I2C0 clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
i2c0_clock_source	I2C0 clock source selection
<i>RCU_I2C0SRC_CKAPB1</i>	CK_APB1 selected as I2C0 source clock
<i>RCU_I2C0SRC_CKSYS</i>	CK_SYS selected as I2C0 source clock
<i>RCU_I2C0SRC_IRC16M</i>	CK_IRC16M selected as I2C0 source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select IRC16M as I2C0 source clock */
```

```
rcu_i2c0_clock_config(RCU_I2C0SRC_IRC16M);
```

rcu_usart0_clock_config

The description of rcu_usart0_clock_config is shown as below:

Table 3-676. Function rcu_usart0_clock_config

Function name	rcu_usart0_clock_config
Function prototype	void rcu_usart0_clock_config(uint32_t usart0_clock_source);

Function descriptions	configure the USART0 clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usart0_clock_source	USART0 clock source selection
<i>RCU_USART0SRC_CKAPB1</i>	CK_APB1 selected as USART0 source clock
<i>RCU_USART0SRC_CKSYS</i>	CK_SYS selected as USART0 source clock
<i>RCU_USART0SRC_CK_LXTAL</i>	CK_LXTAL selected as USART0 source clock
<i>RCU_USART0SRC_CK_IRC16M</i>	CK_IRC16M selected as USART0 source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select APB1 clock as USART0 clock */
```

```
rcu_usart0_clock_config(RCU_USART0SRC_CKAPB1);
```

rcu_usart2_clock_config

The description of rcu_usart2_clock_config is shown as below:

Table 3-677. Function rcu_usart2_clock_config

Function name	rcu_usart2_clock_config
Function prototype	void rcu_usart2_clock_config(uint32_t usart2_clock_source);
Function descriptions	configure the USART2 clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usart2_clock_source	USART2 clock source selection
<i>RCU_USART2SRC_CKAPB1</i>	CK_APB1 selected as USART2 source clock
<i>RCU_USART2SRC_CKSYS</i>	CK_SYS selected as USART2 source clock
<i>RCU_USART2SRC_CK_LXTAL</i>	CK_LXTAL selected as USART2 source clock
<i>RCU_USART2SRC_CK_IRC16M</i>	CK_IRC16M selected as USART2 source clock
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* select CK_APB1 as USART2 source clock */
```

```
rcu_usart2_clock_config(RCU_USART2SRC_CKAPB1);
```

rcu_irc16m_div_config

The description of rcu_irc16m_div_config is shown as below:

Table 3-678. Function rcu_irc16m_div_config

Function name	rcu_irc16m_div_config
Function prototype	void rcu_irc16m_div_config(uint32_t irc16m_div);
Function descriptions	configure IRC16M clock divider factor for system clock
Precondition	-
The called functions	-
Input parameter{in}	
irc16m_div	IRC16M clock divider factor for system clock
<i>RCU_IRC16M_DIVx</i>	IRC16M clock divided by x for system clock (x=1,2,3,...,512)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* IRC16M clock divided by 4 for system clock */
```

```
rcu_irc16m_div_config(RCU_IRC16M_DIV4);
```

rcu_timer_clock_prescaler_config

The description of rcu_timer_clock_prescaler_config is shown as below:

Table 3-679. Function rcu_sdio_div_config

Function name	rcu_timer_clock_prescaler_config
Function prototype	void rcu_timer_clock_prescaler_config(uint32_t timer_clock_prescaler);
Function descriptions	configure the TIMER clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_clock_prescaler	TIMER clock selection
RCU_TIMER_PSC_MU L2	If CK_APBx = CK_AHB or CK_APBx = CK_AHB/2, CK_TIMERx = CK_AHB, else CK_TIMERx = 2 x CK_APBx

RCU_TIMER_PSC_MUL4	If CK_APBx = CK_AHB or CK_APBx = CK_AHB/2 or CK_APBx = CK_AHB/4, CK_TIMERx = CK_AHB, else CK_TIMERx = 4 x CK_APBx
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER clock source */
```

```
rcu_timer_clock_prescaler_config(RCU_TIMER_PSC_MUL4);
```

rcu_lxtal_drive_capability_config

The description of rcu_lxtal_drive_capability_config is shown as below:

Table 3-680. 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
<i>RCU_LXTALDRI_LOWER_DRIVE</i>	lower driving capability
<i>RCU_LXTALDRI_HIGH_DRIVE</i>	high driving capability
<i>RCU_LXTALDRI_HIGHER_DRIVE</i>	higher driving capability
<i>RCU_LXTALDRI_HIGHEST_DRIVE</i>	highest driving capability
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the LXTAL drive capability */
```

```
rcu_lxtal_drive_capability_config (RCU_LXTAL_LOWDRI);
```

rcu_osc_i_stab_wait

The description of rcu_osc_i_stab_wait is shown as below:

Table 3-681. 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 3-632. Enum 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 3-682. 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 3-632. Enum 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 3-683. 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 3-632. Enum 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 3-684. 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 3-632. Enum 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 3-685. 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 3-632. Enum rcu_osci_type_enum
<i>RCU_HXTAL</i>	high speed crystal oscillator(HXTAL)
<i>RCU_LXTAL</i>	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 3-686. 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 3-687. 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_rf_hxtal_clock_monitor_enable

The description of rcu_rf_hxtal_clock_monitor_enable is shown as below:

Table 3-688. Function rcu_rf_hxtal_clock_monitor_enable

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

Example:

```
/* enable the RF HXTAL clock monitor */
rcu_rf_hxtal_clock_monitor_enable();
```

rcu_rf_hxtal_clock_monitor_disable

The description of rcu_rf_hxtal_clock_monitor_disable is shown as below:

Table 3-689. Function rcu_rf_hxtal_clock_monitor_disable

Function name	rcu_rf_hxtal_clock_monitor_disable
----------------------	------------------------------------

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

Example:

```
/* disable the RF HXTAL clock monitor */
rcu_rf_hxtal_clock_monitor_disable();
```

rcu_IRC16M_adjust_value_set

The description of rcu_IRC16M_adjust_value_set is shown as below:

Table 3-690. Function rcu_IRC16M_adjust_value_set

Function name	rcu_IRC16M_adjust_value_set
Function prototype	void rcu_IRC16M_adjust_value_set(uint32_t IRC16M_adjval);
Function descriptions	set the IRC16M adjust value
Precondition	-
The called functions	-
Input parameter{in}	
IRC16M_adjval	IRC16M adjust value, must be between 0 and 0x1F
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the IRC16M adjust value */
rcu_IRC16M_adjust_value_set(0x10);
```

rcu_spread_spectrum_config

The description of rcu_spread_spectrum_config is shown as below:

Table 3-691. Function rcu_spread_spectrum_config

Function name	rcu_spread_spectrum_config
Function prototype	void rcu_spread_spectrum_config(uint32_t spread_spectrum_type, uint32_t modstep, uint32_t modcnt);

Function descriptions	configure the spread spectrum modulation for the main PLL clock
Precondition	-
The called functions	-
Input parameter{in}	
<i>spread_spectrum_type</i>	PLL spread spectrum modulation type select
<i>RCU_SS_TYPE_CENTER</i>	center spread type is selected
<i>RCU_SS_TYPE_DOWN</i>	down spread type is selected
Input parameter{in}	
modstep	configure PLL spread spectrum modulation profile amplitude
<i>uint32_t</i>	0 ~ 0x7FFF, The following criteria must be met: $MODSTEP * MODCNT \leq 2^{15}-1$
Input parameter{in}	
modcnt	configure PLL spread spectrum modulation profile frequency
<i>uint32_t</i>	0 ~ 0x1FFF, The following criteria must be met: $MODSTEP * MODCNT \leq 2^{15}-1$
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure PLL spread_spectrum */
```

```
rcu_spread_spectrum_config(RCU_SS_TYPE_CENTER, 0x0F, 0x0F);
```

rcu_spread_spectrum_enable

The description of rcu_spread_spectrum_enable is shown as below:

Table 3-692. Function rcu_spread_spectrum_enable

Function name	rcu_spread_spectrum_enable
Function prototype	void rcu_spread_spectrum_enable(void);
Function descriptions	enable the PLL spread spectrum modulation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the PLL spread spectrum modulation */
```

```
rcu_spread_spectrum_enable ();
```

rcu_spread_spectrum_disable

The description of rcu_spread_spectrum_disable is shown as below:

Table 3-693. Function rcu_spread_spectrum_disable

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

Example:

```
/* disable the PLL spread spectrum modulation */
```

```
rcu_spread_spectrum_disable();
```

rcu_voltage_key_unlock

The description of rcu_voltage_key_unlock is shown as below:

Table 3-694. Function rcu_voltage_key_unlock

Function name	rcu_voltage_key_unlock
Function prototype	void rcu_voltage_key_unlock(void);
Function descriptions	unlock the voltage key
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*unlock the voltage key */
```

```
rcu_voltage_key_unlock();
```

rcu_deepsleep_voltage_set

The description of rcu_deepsleep_voltage_set is shown as below:

Table 3-695. 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_1	the core voltage is 1.1V in deep-sleep mode
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
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 3-696. 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	Clock frequency, refers to Table 3-633. Enum rcu_clock_freq_enum
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);
```

rcu_security_enable

The description of rcu_security_enable is shown as below:

Table 3-697. Function rcu_security_enable

Function name	rcu_security_enable
Function prototype	void rcu_security_enable(rcu_sec_enum security);
Function descriptions	enable the security attribution
Precondition	-
The called functions	-
Input parameter{in}	
security	clock security attribution, refer to Table 3-634. Enum rcu_sec_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the PLLI2S configuration and status bits security attribution */

rcu_security_enable (RCU_SEC_PLLI2SSEC);
```

rcu_security_disable

The description of rcu_security_disable is shown as below:

Table 3-698. Function rcu_security_disable

Function name	rcu_security_disable
Function prototype	void rcu_security_disable(rcu_sec_enum security);
Function descriptions	disable the security attribution
Precondition	-
The called functions	-
Input parameter{in}	
security	clock security attribution, refer to Table 3-634. Enum rcu_sec_enum
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable the PLLI2S configuration and status bits security attribution */
```

```
rcu_security_disable (RCU_SEC_PLLI2SSEC);
```

rcu_privilege_enable

The description of rcu_privilege_enable is shown as below:

Table 3-699. Function rcu_privilege_enable

Function name	rcu_privilege_enable
Function prototype	void rcu_privilege_enable(void);
Function descriptions	enable the privileged access
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* privileged access enable */
```

```
rcu_privilege_enable ();
```

rcu_privilege_disable

The description of rcu_privilege_disable is shown as below:

Table 3-700. Function rcu_privilege_disable

Function name	rcu_privilege_disable
Function prototype	void rcu_privilege_disable(void);
Function descriptions	disable the privileged access
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* disable the privileged access */
```

```
rcu_privilege_disable ();
```

rcu_flag_get

The description of rcu_flag_get is shown as below:

Table 3-701. 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 peripheral reset flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	the clock stabilization and peripheral reset flags, refer to Table 3-628. Enum 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 3-702. 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 3-703. 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 3-628. Enum rcu_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 3-704. 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);
Function descriptions	clear the interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	clock stabilization and stuck interrupt flags clear, refer to Table 3-630. Enum rcu_int_flag_clear_enum
Output parameter{out}	

Example:

```
rcu_interrupt_flag_clear(RCU_INT_FLAG_HXTALSTB_CLR);
```

rcu_security_flag_get

The description of `rcu_security_flag_get` is shown as below:

Table 3-705. Function rcu_security_flag_get

473

$\}.$

The description of rcu_interrupt_enable is shown as below:

```
rcu_interrupt_enable(RCU_INT_HXTALSTB);
```

rcu_interrupt_disable

The description of rcu_interrupt_disable is shown as below:

Table 3-707. Function rcu_interrupt_disable

Function name	rcu_interrupt_disable
Function prototype	void rcu_interrupt_disable(rcu_int_enum interrupt);
Function descriptions	disable the stabilization interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	clock stabilization interrupt, refer to Table 3-631. Enum rcu_int_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

3.22. 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, two alarms, a prescaler, a divider and the RTC clock configuration register. The RTC registers are listed in chapter [3.22.1](#), the RTC firmware functions are introduced in chapter [3.22.2](#).

3.22.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

Table 3-708. RTC Registers

Registers	Descriptions
RTC_TIME	time of day register
RTC_DATE	date register
RTC_CTL	control register
RTC_ICS	initialization control and status register
RTC_PSC	time prescaler register
RTC_WUT	wakeup timer register
RTC_COSC	coarse calibration register
RTC_ALRM0TD	alarm 0 time and date register
RTC_ALRM1TD	alarm 1 time and date register

Registers	Descriptions
RTC_WPK	write protection key register
RTC_SS	sub second register
RTC_SHIFTCTL	shift function control register
RTC_TTS	time of timestamp register
RTC_DTS	date of timestamp register
RTC_SSTS	sub second of timestamp register
RTC_HRFC	high resolution frequency compensation register
RTC_TAMP	tamper register
RTC_ALARM0SS	alarm 0 sub second register
RTC_ALARM1SS	alarm1 sub second register
RTC_PPM_CTL	privilege protection mode control register
RTC_SPM_CTL	secure protection mode control register
RTC_STAT	status register
RTC_NSMI_STAT	non-secure masked interrupt status register
RTC_SMI_STAT	secure masked interrupt status register
RTC_STATC	status flag clear register
RTC_BKPx(x = 0, 1, 2, ..., 18, 19)	backup register

3.22.2. Descriptions of Peripheral functions

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

Table 3-709. RTC firmware function

Function name	Function description
rtc_deinit	reset most of the RTC registers
rtc_init	initialize RTC registers
rtc_init_mode_enter	enter RTC init mode
rtc_init_mode_exit	exit RTC init mode
rtc_register_sync_wait	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
rtc_current_time_get	get current time and date
rtc_subsecond_get	get current subsecond value
rtc_alarm_config	configure RTC alarm
rtc_alarm_subsecond_config	configure subsecond of RTC alarm
rtc_alarm_get	get RTC alarm
rtc_alarm_subsecond_get	get RTC alarm subsecond
rtc_alarm_enable	enable RTC alarm
rtc_alarm_disable	disable RTC alarm
rtc_timestamp_enable	enable RTC time-stamp
rtc_timestamp_disable	disable RTC time-stamp

Function name	Function description
rtc_timestamp_get	get RTC timestamp time and date
rtc_timestamp_subsecond_get	get RTC time-stamp subsecond
rtc_timestamp_pin_map	RTC time-stamp pin map
rtc_tamper_enable	enable RTC tamper
rtc_tamper_disable	disable RTC tamper
rtc_software_bkp_reset	reset the RTC_BKP registers by software
rtc_tamper_without_bkp_seset	tamperx event does not erase the RTC_BKP registers
rtc_interrupt_enable	enable specified RTC interrupt
rtc_interrupt_disable	disble specified RTC interrupt
rtc_flag_get	get specified flag
rtc_flag_clear	clear specified flag
rtc_nsec_interrupt_flag_get	get specified non-secure interrupt flag
rtc_nsec_interrupt_flag_clear	clear specified non-secure interrupt flag
rtc_sec_interrupt_flag_get	get specified secure interrupt flag
rtc_sec_interrupt_flag_clear	clear specified secure interrupt flag
rtc_output_pad_select	select the RTC output pin pad
rtc_alarm_output_config	configure RTC alarm output source
rtc_calibration_output_config	configure RTC calibration output source
rtc_hour_adjust	ajust the daylight saving time by adding or subtracting one hour from the current time
rtc_second_adjust	ajust RTC second or subsecond value of current time
rtc_bypass_shadow_enable	enable RTC bypass shadow registers function
rtc_bypass_shadow_disable	disable RTC bypass shadow registers function
rtc_refclock_detection_enable	enable RTC reference clock detection function
rtc_refclock_detection_disable	disable RTC reference clock detection function
rtc_wakeup_enable	enable RTC wakeup timer
rtc_wakeup_disable	disable RTC wakeup timer
rtc_wakeup_clock_set	set auto wakeup timer clock
rtc_wakeup_timer_set	set auto wakeup timer value
rtc_wakeup_timer_get	get auto wakeup timer value
rtc_smooth_calibration_config	configure RTC smooth calibration
rtc_coarse_calibration_enable	enable RTC coarse calibration
rtc_coarse_calibration_disable	disable RTC coarse calibration
rtc_coarse_calibration_config	configure RTC coarse calibration direction and step
rtc_pri_pro_enable	enable RTC privilege protection mode
rtc_pri_pro_disable	disable RTC privilege protection mode
rtc_sec_pro_enable	enable RTC secure protection mode
rtc_sec_pro_disable	disable RTC secure protection mode
rtc_bkp_zonea_sec_pro_set	set the RTC_BKP secure protection zonea tail value
rtc_bkp_zoneb_sec_pro_set	set the RTC_BKP secure protection zoneb tail value
rtc_bkp_zoneb_sec_pro_check	check the RTC_BKP secure protection zoneb is valid or not

Structure rtc_parameter_struct

Table 3-710. Structure rtc_parameter_struct

Member name	Function description
year	RTC year value: 0x0 - 0x99(BCD format)
month	RTC month value
date	RTC date value: 0x1 - 0x31(BCD format)
day_of_week	RTC weekday value
hour	RTC hour value
minute	RTC minute value: 0x0 - 0x59(BCD format)
second	RTC second value: 0x0 - 0x59(BCD format)
factor_asyn	RTC asynchronous prescaler value: 0x0 - 0x7F
factor_syn	RTC synchronous prescaler value: 0x0 - 0x7FFF
am_pm	RTC AM/PM value
display_format	RTC time notation

Structure rtc_alarm_struct

Table 3-711. Structure rtc_alarm_struct

Member name	Function description
alarm_mask	RTC alarm mask
weekday_or_date	specify RTC alarm is on date or weekday
alarm_day	RTC alarm date or weekday value
alarm_hour	RTC alarm hour value
alarm_minute	RTC alarm minute value: 0x0 - 0x59(BCD format)
alarm_second	RTC alarm second value: 0x0 - 0x59(BCD format)
am_pm	RTC alarm AM/PM value

Structure rtc_timestamp_struct

Table 3-712. Structure rtc_timestamp_struct

Member name	Function description
timestamp_month	RTC time-stamp month value
timestamp_date	RTC time-stamp date value: 0x1 - 0x31(BCD format)
timestamp_day	RTC time-stamp weekday value
timestamp_hour	RTC time-stamp hour value
timestamp_minute	RTC time-stamp minute value: 0x0 - 0x59(BCD format)
timestamp_second	RTC time-stamp second value: 0x0 - 0x59(BCD format)
am_pm	RTC time-stamp AM/PM value

Structure rtc_tamper_struct

Table 3-713. Structure rtc_tamper_struct

Member name	Function description
-------------	----------------------

tamper_source	RTC tamper source
tamper_trigger	RTC tamper trigger
tamper_filter	RTC tamper consecutive samples needed during a voltage level detection
tamper_sample_frequency	RTC tamper sampling frequency during a voltage level detection
tamper_precharge_enable	RTC tamper precharge feature during a voltage level detection
tamper_precharge_time	RTC tamper precharge duration if precharge feature is enabled
tamper_with_timestamp	RTC tamper time-stamp feature

rtc_deinit

The description of rtc_deinit is shown as below:

Table 3-714. Function rtc_deinit

Function name	rtc_deinit
Function prototype	ErrStatus rtc_deinit(void);
Function descriptions	reset most of the RTC registers
Precondition	-
The called functions	rcu_periph_reset_enable/ rcu_periph_reset_disable -
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* reset most of the RTC registers*/
```

```
ErrStatus error_status = rtc_deinit();
```

rtc_init

The description of rtc_init is shown as below:

Table 3-715. Function rtc_init

Function name	rtc_init
Function prototype	ErrStatus rtc_init(rtc_parameter_struct* rtc_initpara_struct);
Function descriptions	initialize RTC registers
Precondition	-
The called functions	-
Input parameter{in}	

rtc_initpara_struct	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure Table 3-710. Structure rtc_parameter_struct
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```

/* initialize RTC registers */

rtc_parameter_struct rtc_initpara;

rtc_interrupt_disable(RTC_INT_SECOND);

rtc_initpara.factor_asyn = prescaler_a;

rtc_initpara.factor_syn = prescaler_s;

rtc_initpara.year = 0x16;

rtc_initpara.day_of_week = RTC_SATURDAY;

rtc_initpara.month = RTC_APR;

rtc_initpara.date = 0x30;

rtc_initpara.display_format = RTC_24HOUR;

rtc_initpara.am_pm = RTC_AM;

rtc_init(&rtc_initpara);

```

rtc_init_mode_enter

The description of rtc_init_mode_enter is shown as below:

Table 3-716. Function rtc_init_mode_enter

Function name	rtc_init_mode_enter
Function prototype	ErrStatus rtc_init_mode_enter(void);
Function descriptions	enter RTC init mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/*enter RTC init mode*/
```

```
ErrStatus error_status = rtc_init_mode_enter ();
```

rtc_init_mode_exit

The description of rtc_init_mode_exit is shown as below:

Table 3-717. Function rtc_init_mode_exit

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

Example:

```
/*exit RTC init mode*/
```

```
rtc_init_mode_exit ();
```

rtc_register_sync_wait

The description of rtc_register_sync_wait is shown as below:

Table 3-718. Function rtc_register_sync_wait

Function name	rtc_register_sync_wait
Function prototype	ErrStatus rtc_register_sync_wait(void);
Function descriptions	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/*wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the
shadow registers are updated*/
```

```
ErrStatus error_status = rtc_register_sync_wait ();
```

rtc_current_time_get

The description of rtc_current_time_get is shown as below:

Table 3-719. Function rtc_current_time_get

Function name	rtc_current_time_get
Function prototype	void rtc_current_time_get(rtc_parameter_struct* rtc_initpara_struct);
Function descriptions	get current time and date
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
rtc_initpara_struct	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure Table 3-710. Structure rtc_parameter_struct
Return value	
-	-

Example:

```
/*get current time and date*/

rtc_parameter_struct rtc_initpara_struct;

rtc_current_time_get (&rtc_initpara_struct);
```

rtc_subsecond_get

The description of rtc_subsecond_get is shown as below:

Table 3-720. Function rtc_subsecond_get

Function name	rtc_subsecond_get
Function prototype	uint32_t rtc_subsecond_get(void);
Function descriptions	get current subsecond value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

uint32_t	current subsecond value(0x00-0xFFFF)
-----------------	--------------------------------------

Example:

```
/*get current subsecond value*/
```

```
uint32_t sub_second = rtc_subsecond_get();
```

rtc_alarm_config

The description of rtc_alarm_config is shown as below:

Table 3-721. Function rtc_alarm_config

Function name	rtc_alarm_config
Function prototype	void rtc_alarm_config(uint8_t rtc_alarm, rtc_alarm_struct* rtc_alarm_time)
Function descriptions	configure RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	select alarm
<i>RTC_ALARM0</i>	alarm 0
<i>RTC_ALARM1</i>	alarm 1
Input parameter{in}	
rtc_alarm_time	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure Table 3-711. Structure rtc_alarm_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*rtc_alarm_config*/
```

```
rtc_alarm_struct rtc_alarm_time;
```

```
rtc_alarm_config(RTC_ALARM0,&rtc_alarm_time);
```

rtc_alarm_subsecond_config

The description of rtc_alarm_subsecond_config is shown as below:

Table 3-722. Function rtc_alarm_subsecond_config

Function name	rtc_alarm_subsecond_config
Function prototype	void rtc_alarm_subsecond_config(uint8_t rtc_alarm, uint32_t mask_subsecond, uint32_t subsecond)
Function descriptions	configure subsecond of RTC alarm

Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	select alarm
<i>RTC_ALARM0</i>	alarm 0
<i>RTC_ALARM1</i>	alarm 1
Input parameter{in}	
mask_subsecond	alarm subsecond mask
<i>RTC_MASKSSC_0_14</i>	mask alarm subsecond configuration
<i>RTC_MASKSSC_1_14</i>	mask RTC_ALRM0SS_SSC[14:1], and RTC_ALRM0SS_SSC[0] is to be compared
<i>RTC_MASKSSC_2_14</i>	mask RTC_ALRM0SS_SSC[14:2], and RTC_ALRM0SS_SSC[1:0] is to be compared
<i>RTC_MASKSSC_3_14</i>	mask RTC_ALRM0SS_SSC[14:3], and RTC_ALRM0SS_SSC[2:0] is to be compared
<i>RTC_MASKSSC_4_14</i>	mask RTC_ALRM0SS_SSC[14:4], and RTC_ALRM0SS_SSC[3:0] is to be compared
<i>RTC_MASKSSC_5_14</i>	mask RTC_ALRM0SS_SSC[14:5], and RTC_ALRM0SS_SSC[4:0] is to be compared
<i>RTC_MASKSSC_6_14</i>	mask RTC_ALRM0SS_SSC[14:6], and RTC_ALRM0SS_SSC[5:0] is to be compared
<i>RTC_MASKSSC_7_14</i>	mask RTC_ALRM0SS_SSC[14:7], and RTC_ALRM0SS_SSC[6:0] is to be compared
<i>RTC_MASKSSC_8_14</i>	mask RTC_ALRM0SS_SSC[14:8], and RTC_ALRM0SS_SSC[7:0] is to be compared
<i>RTC_MASKSSC_9_14</i>	mask RTC_ALRM0SS_SSC[14:9], and RTC_ALRM0SS_SSC[8:0] is to be compared
<i>RTC_MASKSSC_10_14</i>	mask RTC_ALRM0SS_SSC[14:10], and RTC_ALRM0SS_SSC[9:0] is to be compared
<i>RTC_MASKSSC_11_14</i>	mask RTC_ALRM0SS_SSC[14:11], and RTC_ALRM0SS_SSC[10:0] is to be compared
<i>RTC_MASKSSC_12_14</i>	mask RTC_ALRM0SS_SSC[14:12], and RTC_ALRM0SS_SSC[11:0] is to be compared
<i>RTC_MASKSSC_13_14</i>	mask RTC_ALRM0SS_SSC[14:13], and RTC_ALRM0SS_SSC[12:0] is to be compared
<i>RTC_MASKSSC_14</i>	mask RTC_ALRM0SS_SSC[14], and RTC_ALRM0SS_SSC[13:0] is to be compared
<i>RTC_MASKSSC_NONE</i>	mask none, and RTC_ALRM0SS_SSC[14:0] is to be compared
Input parameter{in}	
subsecond	alarm subsecond value(0x000 - 0x7FFF)
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/*configure subsecond of RTC alarm0*/
```

```
rtc_subsecond_config (RTC_ALARM0,RTC_MASKSSC_9_14, 0x7FFF);
```

rtc_alarm_get

The description of rtc_alarm_get is shown as below:

Table 3-723. Function rtc_alarm_get

Function name	rtc_alarm_get
Function prototype	void rtc_alarm_get(rtc_alarm_struct* rtc_alarm_time);
Function descriptions	get RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	select alarm
<i>RTC_ALARM0</i>	alarm 0
<i>RTC_ALARM1</i>	alarm 1
Output parameter{out}	
rtc_alarm_time	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure Table 3-711. Structure rtc_alarm_struct
Return value	
-	-

Example:

```
rtc_alarm_struct rtc_alarm_time;
```

```
rtc_alarm_get(RTC_ALARM0,&rtc_alarm_time);
```

rtc_alarm_subsecond_get

The description of rtc_alarm_subsecond_get is shown as below:

Table 3-724. Function rtc_alarm_subsecond_get

Function name	rtc_alarm_subsecond_get
Function prototype	uint32_t rtc_alarm_subsecond_get(uint8_t rtc_alarm)
Function descriptions	get RTC alarm subsecond
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	select alarm

<i>RTC_ALARM0</i>	alarm 0
<i>RTC_ALARM1</i>	alarm 1
Output parameter{out}	
-	-
Return value	
uint32_t	RTC alarm subsecond value(0x0-0x7FFF)

Example:

```
/*get RTC alarm0 subsecond*/
```

```
uint32_t subsecond = rtc_alarm_subsecond_get(RTC_ALARM0);
```

rtc_alarm_enable

The description of rtc_alarm_enable is shown as below:

Table 3-725. Function rtc_alarm_enable

Function name	rtc_alarm_enable
Function prototype	void rtc_alarm_enable(uint8_t rtc_alarm)
Function descriptions	enable RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	select alarm
<i>RTC_ALARM0</i>	alarm 0
<i>RTC_ALARM1</i>	alarm 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*enable RTC alarm0*/
```

```
rtc_alarm_enable(RTC_ALARM0);
```

rtc_alarm_disable

The description of rtc_alarm_disable is shown as below:

Table 3-726. Function rtc_alarm_disable

Function name	rtc_alarm_disable
Function prototype	ErrStatus rtc_alarm_disable(uint8_t rtc_alarm)
Function descriptions	disable RTC alarm
Precondition	-
The called functions	-

Input parameter{in}	
rtc_alarm	select alarm
<i>RTC_ALARM0</i>	alarm 0
<i>RTC_ALARM1</i>	alarm 1
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/*disable RTC alarm0*/
```

```
ErrStatus error_status = rtc_alarm_disable(RTC_ALARM0);
```

rtc_timestamp_enable

The description of `rtc_timestamp_enable` is shown as below:

Table 3-727. Function `rtc_timestamp_enable`

Function name	<code>rtc_timestamp_enable</code>
Function prototype	<code>void rtc_timestamp_enable(uint32_t edge);</code>
Function descriptions	enable RTC time-stamp
Precondition	-
The called functions	-
Input parameter{in}	
edge	specify which edge to detect of time-stamp
<i>RTC_TIMESTAMP_RISING_EDGE</i>	rising edge is valid event edge for timestamp event
<i>RTC_TIMESTAMP_FALLING_EDGE</i>	falling edge is valid event edge for timestamp event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*enable RTC time-stamp*/
```

```
rtc_timestamp_enable (RTC_TIMESTAMP_RISING_EDGE);
```

rtc_timestamp_disable

The description of `rtc_timestamp_disable` is shown as below:

Table 3-728. Function `rtc_timestamp_disable`

Function name	<code>rtc_timestamp_disable</code>
----------------------	------------------------------------

Function prototype	void rtc_timestamp_disable(void);
Function descriptions	disable RTC time-stamp
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC time-stamp */
rtc_timestamp_disable ();
```

rtc_timestamp_get

The description of rtc_timestamp_get is shown as below:

Table 3-729. Function rtc_timestamp_get

Function name	rtc_timestamp_get
Function prototype	void rtc_timestamp_get(rtc_timestamp_struct* rtc_timestamp);
Function descriptions	get RTC timestamp time and date
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
rtc_timestamp	Pointer to a rtc_timestamp_struct structure which contains parameters for RTC time-stamp configuration, the structure members can refer to members of the structure Table 3-713. Structure rtc_tamper_struct
Return value	
-	-

Example:

```
/* get RTC timestamp time and date */
rtc_timestamp_struct rtc_timestamp;
rtc_timestamp_get(& rtc_timestamp);
```

rtc_timestamp_subsecond_get

The description of rtc_timestamp_subsecond_get is shown as below:

Table 3-730. Function rtc_timestamp_subsecond_get

Function name	rtc_timestamp_subsecond_get
Function prototype	uint32_t rtc_timestamp_subsecond_get(void);
Function descriptions	get RTC time-stamp subsecond
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	RTC time-stamp subsecond value

Example:

```
/* get RTC time-stamp subsecond */
```

```
uint32_t subsecond = rtc_timestamp_subsecond_get();
```

rtc_tamper_enable

The description of rtc_tamper_enable is shown as below:

Table 3-731. Function rtc_tamper_enable

Function name	rtc_tamper_enable
Function prototype	void rtc_tamper_enable(rtc_tamper_struct* rtc_tamper);
Function descriptions	enable RTC tamper
Precondition	-
The called functions	-
Input parameter{in}	
rtc_tamper	pointer to a rtc_tamper_struct structure which contains parameters for RTC tamper configuration, the structure members can refer to members of the structure Table 3-713. Structure rtc_tamper_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC tamper */
```

```
rtc_tamper_struct rtc_tamper
```

```
rtc_tamper_enable(& rtc_tamper);
```

rtc_tamper_disable

The description of rtc_tamper_disable is shown as below:

Table 3-732. Function rtc_tamper_disable

Function name	rtc_tamper_disable
Function prototype	void rtc_tamper_disable(uint32_t source);
Function descriptions	disable RTC tamper
Precondition	-
The called functions	-
Input parameter{in}	
source	specify which tamper source to be disabled
<i>RTC_TAMPER0</i>	RTC tamper0
<i>RTC_TAMPER1</i>	RTC tamper1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC tamper */
rtc_tamper_disable(RTC_TAMPER0);
```

rtc_software_bkp_reset

The description of rtc_software_bkp_reset is shown as below:

Table 3-733. Function rtc_software_bkp_reset

Function name	rtc_software_bkp_reset
Function prototype	void rtc_software_bkp_reset(void)
Function descriptions	reset the RTC_BKP registers by software
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the RTC_BKP registers by software */
rtc_software_bkp_reset ();
```

rtc_tamper_without_bkp_seset

The description of rtc_tamper_without_bkp_seset is shown as below:

Table 3-734. Function rtc_tamper_without_bkp_seset

Function name	rtc_tamper_without_bkp_seset
Function prototype	void rtc_tamper_without_bkp_seset(uint32_t ne_source)
Function descriptions	tamperx event does not erase the RTC_BKP registers
Precondition	-
The called functions	-
Input parameter{in}	
ne_source	specify which tamper source will not trigger the RTC_BKP registers reset
<i>RTC_TAMPXNOER_NONE</i>	both tamper0 and tamper1 event will trigger RTC_BKP registers reset
<i>RTC_TAMPXNOER_TP0</i>	tamper0 event will not trigger RTC_BKP registers reset
<i>RTC_TAMPXNOER_TP1</i>	tamper1 event will not trigger RTC_BKP registers reset
<i>RTC_TAMPXNOER_TP0_TP1</i>	neither tamper0 nor tamper1 event will trigger RTC_BKP registers reset
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set tamper0 will not trigger RTC_BKP registers */
rtc_tamper_without_bkp_seset(RTC_TAMPXNOER_TP0);
```

rtc_interrupt_enable

The description of rtc_interrupt_enable is shown as below:

Table 3-735. Function rtc_interrupt_enable

Function name	rtc_interrupt_enable
Function prototype	void rtc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable specified RTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify which interrupt source to be enabled
<i>RTC_INT_TIMESTAMP</i>	timestamp interrupt
<i>RTC_INT_ALARM0</i>	alarm0 interrupt
<i>RTC_INT_ALARM1</i>	alarm1 interrupt

<i>RTC_INT_WAKEUP</i>	wakeup timer interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable specified RTC interrupt*/
```

```
rtc_interrupt_enable(RTC_INT_ALARM0);
```

rtc_interrupt_disable

The description of `rtc_interrupt_disable` is shown as below:

Table 3-736. Function `rtc_interrupt_disable`

Function name	<code>rtc_interrupt_disable</code>
Function prototype	<code>void rtc_interrupt_disable(uint32_t interrupt);</code>
Function descriptions	disable specified RTC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify which RTC interrupt to disable
<i>RTC_INT_TIMESTAMP</i>	timestamp interrupt
<i>RTC_INT_ALARM0</i>	alarm0 interrupt
<i>RTC_INT_ALARM1</i>	alarm1 interrupt
<i>RTC_INT_WAKEUP</i>	wakeup timer interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable specified RTC interrupt */
```

```
rtc_interrupt_disable(RTC_INT_ALARM0);
```

rtc_flag_get

The description of `rtc_flag_get` is shown as below:

Table 3-737. Function `rtc_flag_get`

Function name	<code>rtc_flag_get</code>
Function prototype	<code>FlagStatus rtc_flag_get(uint32_t flag);</code>
Function descriptions	check specified flag
Precondition	-

The called functions	-
Input parameter{in}	
flag	specify which flag to check
<i>RTC_FLAG_SCP</i>	smooth calibration pending flag
<i>RTC_FLAG_TP1</i>	RTC tamper 1 detected flag
<i>RTC_FLAG_TP0</i>	RTC tamper 0 detected flag
<i>RTC_FLAG_TSOVR</i>	time-stamp overflow event flag
<i>RTC_FLAG_TS</i>	time-stamp event flag
<i>RTC_FLAG_ALARM0</i>	alarm0 event flag
<i>RTC_FLAG_ALARM1</i>	alarm1 event flag
<i>RTC_FLAG_WT</i>	wakeup timer occurs flag
<i>RTC_FLAG_INIT</i>	initialization state flag
<i>RTC_FLAG_RSYN</i>	register synchronization flag
<i>RTC_FLAG_YCM</i>	year configuration mark status flag
<i>RTC_FLAG_SOP</i>	shift function operation pending flag
<i>RTC_FLAG_ALARM0W</i>	alarm0 configuration can be written flag
<i>RTC_FLAG_ALARM1W</i>	alarm1 configuration can be written flag
<i>RTC_FLAG_WTW</i>	wakeup timer can be written flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check time-stamp event flag */
```

```
FlagStatus = rtc_flag_get(RTC_FLAG_TS);
```

rtc_flag_clear

The description of rtc_flag_clear is shown as below:

Table 3-738. Function rtc_flag_clear

Function name	rtc_flag_clear
Function prototype	void rtc_flag_clear(uint32_t flag);
Function descriptions	clear specified flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag to clear
<i>RTC_FLAG_TP1</i>	RTC tamper 1 detected flag
<i>RTC_FLAG_TP0</i>	RTC tamper 0 detected flag
<i>RTC_FLAG_TSOVR</i>	time-stamp overflow event flag
<i>RTC_FLAG_TS</i>	time-stamp event flag

<i>RTC_FLAG_WT</i>	wakeup timer occurs flag
<i>RTC_FLAG_ALARM0</i>	alarm0 occurs flag
<i>RTC_FLAG_ALARM1</i>	alarm1 occurs flag
<i>RTC_FLAG_RSYN</i>	register synchronization flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear time-stamp event flag */
```

```
rtc_flag_clear (RTC_FLAG_TS);
```

rtc_nsec_interrupt_flag_get

The description of `rtc_nsec_interrupt_flag_get` is shown as below:

Table 3-739. Function `rtc_nsec_interrupt_flag_get`

Function name	<code>rtc_nsec_interrupt_flag_get</code>
Function prototype	<code>FlagStatus rtc_nsec_interrupt_flag_get(uint32_t int_flag);</code>
Function descriptions	get specified non-secure interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	specify which flag to clear
<i>RTC_NSMI_STAT_ALRMONSMF</i>	alarm0 non-secure interrupt masked flag
<i>RTC_NSMI_STAT_ALRM1NSMF</i>	alarm1 non-secure interrupt masked flag
<i>RTC_NSMI_STAT_WTNSMF</i>	wakeup timer non-secure interrupt masked flag
<i>RTC_NSMI_STAT_TSNSMF</i>	time-stamp non-secure interrupt masked flag
<i>RTC_NSMI_STAT_TSOVRNSMF</i>	time-stamp overflow non-secure interrupt masked flag
<i>RTC_NSMI_STAT_TP0NSMF</i>	RTC_TAMP0 non-secure interrupt masked flag
<i>RTC_NSMI_STAT_TP1NSMF</i>	RTC_TAMP1 non-secure interrupt masked flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get alarm0 non-secure interrupt flag */

rtc_nsec_interrupt_flag_get(RTC_NSMI_STAT_ALRM0NSMF);
```

rtc_nsec_interrupt_flag_clear

The description of rtc_nsec_interrupt_flag_clear is shown as below:

Table 3-740. Function rtc_nsec_interrupt_flag_clear

Function name	rtc_nsec_interrupt_flag_clear
Function prototype	void rtc_nsec_interrupt_flag_clear(uint32_t int_flag)
Function descriptions	clear specified non-secure interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	specify which flag to clear
<i>RTC_NSMI_STAT_ALRM0NSMF</i>	alarm0 non-secure interrupt masked flag
<i>RTC_NSMI_STAT_ALRM1NSMF</i>	alarm1 non-secure interrupt masked flag
<i>RTC_NSMI_STAT_WTNSMF</i>	wakeup timer non-secure interrupt masked flag
<i>RTC_NSMI_STAT_TSNSMF</i>	time-stamp non-secure interrupt masked flag
<i>RTC_NSMI_STAT_TSOVRNSMF</i>	time-stamp overflow non-secure interrupt masked flag
<i>RTC_NSMI_STAT_TPO0NSMF</i>	RTC_TAMP0 non-secure interrupt masked flag
<i>RTC_NSMI_STAT_TPO1NSMF</i>	RTC_TAMP1 non-secure interrupt masked flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear alarm0 non-secure interrupt flag */

rtc_nsec_interrupt_flag_clear(RTC_NSMI_STAT_ALRM0NSMF);
```

rtc_sec_interrupt_flag_get

The description of rtc_sec_interrupt_flag_get is shown as below:

Table 3-741. Function `rtc_sec_interrupt_flag_get`

Function name	<code>rtc_sec_interrupt_flag_get</code>
Function prototype	<code>FlagStatus rtc_sec_interrupt_flag_get(uint32_t int_flag);</code>
Function descriptions	get specified secure interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	specify which flag to get
<code>RTC_SMI_STAT_ALR M0SMF</code>	alarm0 secure interrupt masked flag
<code>RTC_SMI_STAT_ALR M1SMF</code>	alarm1 secure interrupt masked flag
<code>RTC_SMI_STAT_WTS MF</code>	wakeup timer secure interrupt masked flag
<code>RTC_SMI_STAT_TSS MF</code>	time-stamp secure interrupt masked flag
<code>RTC_SMI_STAT_TSO VRSMF</code>	time-stamp overflow secure interrupt masked flag
<code>RTC_SMI_STAT_TP0S MF</code>	RTC_TAMP0 secure interrupt masked flag
<code>RTC_SMI_STAT_TP1S MF</code>	RTC_TAMP1 secure interrupt masked flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get alarm0 secure interrupt flag */
rtc_sec_interrupt_flag_get(RTC_SMI_STAT_ALRM0SMF);
```

`rtc_sec_interrupt_flag_clear`

The description of `rtc_sec_interrupt_flag_clear` is shown as below:

Table 3-742. Function `rtc_nsec_interrupt_flag_clear`

Function name	<code>rtc_sec_interrupt_flag_clear</code>
Function prototype	<code>void rtc_sec_interrupt_flag_clear(uint32_t int_flag);</code>
Function descriptions	clear specified secure interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	specify which flag to clear

<i>RTC_SMI_STAT_ALR M0SMF</i>	alarm0 secure interrupt masked flag
<i>RTC_SMI_STAT_ALR M1SMF</i>	alarm1 secure interrupt masked flag
<i>RTC_SMI_STAT_WTS MF</i>	wakeup timer secure interrupt masked flag
<i>RTC_SMI_STAT_TSS MF</i>	time-stamp secure interrupt masked flag
<i>RTC_SMI_STAT_TSO VRSMF</i>	time-stamp overflow secure interrupt masked flag
<i>RTC_SMI_STAT_TP0S MF</i>	RTC_TAMP0 secure interrupt masked flag
<i>RTC_SMI_STAT_TP1S MF</i>	RTC_TAMP1 secure interrupt masked flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear alarm0 secure interrupt flag */
```

```
rtc_sec_interrupt_flag_clear(RTC_SMI_STAT_ALRM0SMF);
```

rtc_output_pad_select

The description of rtc_output_pad_select is shown as below:

Table 3-743. Function rtc_output_pad_select

Function name	rtc_output_pad_select
Function prototype	void rtc_output_pad_select(uint32_t pad);
Function descriptions	select the RTC output pin pad
Precondition	-
The called functions	-
Input parameter{in}	
pad	specify the rtc output pad is PC15 or not
<i>RTC_OUT_PC15</i>	the rtc output pad is PC15
<i>RTC_OUT_PA3_PA8</i>	the rtc output pad is PA3 or PA8 according to the AFIO configuration
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select the rtc output pad is PC15 */
```

```
rtc_output_pad_select(RTC_OUT_PC15);
```

rtc_alarm_output_config

The description of rtc_alarm_output_config is shown as below:

Table 3-744. Function rtc_alarm_output_config

Function name	rtc_alarm_output_config
Function prototype	void rtc_alarm_output_config(uint32_t source, uint32_t mode);
Function descriptions	configure rtc alarm output source
Precondition	-
The called functions	-
Input parameter{in}	
source	specify signal to output
RTC_ALARM0_HIGH	when the alarm0 flag is set, the output pin is high
RTC_ALARM0_LOW	when the alarm0 flag is set, the output pin is low
RTC_ALARM1_HIGH	when the alarm1 flag is set, the output pin is high
RTC_ALARM1_LOW	when the alarm1 flag is set, the output pin is low
RTC_WAKEUP_HIGH	when the wakeup flag is set, the output pin is high
RTC_WAKEUP_LOW	when the wakeup flag is set, the output pin is low
Input parameter{in}	
mode	specify the output pin mode when output alarm signal or auto wakeup signal
RTC_ALARM_OUTPUT_OD	open drain mode
RTC_ALARM_OUTPUT_PP	push pull mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure rtc alternate output source */
```

```
rtc_alarm_output_config(RTC_ALARM0_LOW, RTC_ALARM_OUTPUT_PP);
```

rtc_calibration_config

The description of rtc_calibration_config is shown as below:

Table 3-745. Function rtc_calibration_config

Function name	rtc_calibration_config
Function prototype	void rtc_calibration_output_config(uint32_t source);
Function descriptions	configure rtc calibration output source

Precondition	-
The called functions	-
Input parameter{in}	
source	specify signal to output
<i>RTC_CALIBRATION_5</i> <i>12HZ</i>	when the LSE frequency is 32768Hz and the RTC_PSC is the default value, output 512Hz signal
<i>RTC_CALIBRATION_1</i> <i>HZ</i>	when the LSE frequency is 32768Hz and the RTC_PSC is the default value, output 1Hz signal
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure rtc calibration output source */
rtc_calibration_output_config(RTC_CALIBRATION_1HZ);
```

rtc_hour_adjust

The description of rtc_hour_adjust is shown as below:

Table 3-746. Function rtc_hour_adjust

Function name	rtc_hour_adjust
Function prototype	void rtc_hour_adjust(uint32_t operation);
Function descriptions	adjust the daylight saving time by adding or subtracting one hour from the current time
Precondition	-
The called functions	-
Input parameter{in}	
operation	hour adjustment operation
<i>RTC_CTL_A1H</i>	add one hour
<i>RTC_CTL_S1H</i>	subtract one hour
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* adjust the daylight saving time by adding one hour from the current time */
rtc_hour_adjust(RTC_CTL_A1H);
```

rtc_second_adjust

The description of rtc_second_adjust is shown as below:

Table 3-747. Function rtc_second_adjust

Function name	rtc_second_adjust
Function prototype	ErrStatus rtc_second_adjust(uint32_t add, uint32_t minus);
Function descriptions	adjust RTC second or subsecond value of current time
Precondition	-
The called functions	-
Input parameter{in}	
add	add 1s to current time or not
<i>RTC_SHIFT_ADD1S_RESET</i>	no effect
<i>RTC_SHIFT_ADD1S_SET</i>	add 1s to current time
Input parameter{in}	
minus	number of subsecond to minus from current time(0x0 - 0x7FFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* adjust RTC second or subsecond value of current time */
```

```
ErrStatus error_status = rtc_second_adjust(RTC_SHIFT_ADD1S_SET, 0);
```

rtc_bypass_shadow_enable

The description of rtc_bypass_shadow_enable is shown as below:

Table 3-748. Function rtc_bypass_shadow_enable

Function name	rtc_bypass_shadow_enable
Function prototype	void rtc_bypass_shadow_enable(void);
Function descriptions	enable RTC bypass shadow registers function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:


```
/* enable RTC bypass shadow registers function */
```

```
rtc_bypass_shadow_enable();
```

rtc_bypass_shadow_disable

The description of rtc_bypass_shadow_disable is shown as below:

Table 3-749. Function rtc_bypass_shadow_disable

Function name	rtc_bypass_shadow_disable
Function prototype	void rtc_bypass_shadow_disable (void);
Function descriptions	disable RTC bypass shadow registers function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC bypass shadow registers function */
```

```
rtc_bypass_shadow_disable ();
```

rtc_refclock_detection_enable

The description of rtc_refclock_detection_enable is shown as below:

Table 3-750. Function rtc_refclock_detection_enable

Function name	rtc_refclock_detection_enable
Function prototype	ErrStatus rtc_refclock_detection_enable(void);
Function descriptions	enable RTC reference clock detection function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* enable RTC reference clock detection function*/
```

```
ErrStatus error_status = rtc_refclock_detection_enable();
```

rtc_refclock_detection_disable

The description of rtc_refclock_detection_disable is shown as below:

Table 3-751. Function rtc_refclock_detection_disable

Function name	rtc_refclock_detection_disable
Function prototype	ErrStatus rtc_refclock_detection_disable(void);
Function descriptions	disable RTC reference clock detection function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* disable RTC reference clock detection function */
```

```
ErrStatus error_status = rtc_refclock_detection_disable ();
```

rtc_wakeup_enable

The description of rtc_refclock_detection_disable is shown as below:

Table 3-752. Function rtc_wakeup_enable

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

Example:

```
/* enable RTC auto wakeup function */
```

```
rtc_wakeup_enable();
```

rtc_wakeup_disable

The description of rtc_wakeup_disable is shown as below:

Table 3-753. Function rtc_wakeup_disable

Function name	rtc_wakeup_disable
Function prototype	ErrStatus rtc_wakeup_disable(void);
Function descriptions	disable RTC auto wakeup function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* disable RTC auto wakeup function */
ErrStatus error_status = rtc_wakeup_disable ();
```

rtc_wakeup_clock_set

The description of rtc_wakeup_clock_set is shown as below:

Table 3-754. Function rtc_wakeup_clock_set

Function name	rtc_wakeup_clock_set
Function prototype	ErrStatus rtc_wakeup_clock_set(uint8_t wakeup_clock);
Function descriptions	set RTC auto wakeup timer clock
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_clock	wakeup timer clock is RTC clock divided factor
WAKEUP_RTCK_DIV 16	RTC auto wakeup timer clock is RTC clock divided by 16
WAKEUP_RTCK_DIV 8	RTC auto wakeup timer clock is RTC clock divided by 8
WAKEUP_RTCK_DIV 4	RTC auto wakeup timer clock is RTC clock divided by 4
WAKEUP_RTCK_DIV 2	RTC auto wakeup timer clock is RTC clock divided by 2
WAKEUP_CKSPRE	RTC auto wakeup timer clock is ckspre
WAKEUP_CKSPRE_2 EXP16	RTC auto wakeup timer clock is ckspre and wakeup timer add 2exp16

Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* disable RTC auto wakeup function */
```

```
ErrStatus error_status = rtc_wakeup_clock_set(WAKEUP_RTCK_DIV8);
```

rtc_wakeup_timer_set

The description of rtc_wakeup_timer_set is shown as below:

Table 3-755. Function rtc_wakeup_timer_set

Function name	rtc_wakeup_timer_set
Function prototype	ErrStatus rtc_wakeup_timer_set(uint16_t wakeup_timer);
Function descriptions	set wakeup timer value
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_timer	wakeup timer value
uint16_t	0x0000-0xffff
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* set wakeup timer value */
```

```
ErrStatus error_status = rtc_wakeup_timer_set(0XFFEE);
```

rtc_wakeup_timer_get

The description of rtc_wakeup_timer_set is shown as below:

Table 3-756. Function rtc_wakeup_timer_get

Function name	rtc_wakeup_timer_get
Function prototype	uint16_t rtc_wakeup_timer_get(void);
Function descriptions	set wakeup timer value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
uint16_t	0-0xFFFF

Example:

```
/* get wakeup timer value */
```

```
uint32_t wakeup_time = rtc_wakeup_timer();
```

rtc_smooth_calibration_config

The description of rtc_smooth_calibration_config is shown as below:

Table 3-757. rtc_smooth_calibration_config

Function name	rtc_smooth_calibration_config
Function prototype	ErrStatus rtc_smooth_calibration_config(uint32_t window, uint32_t plus, uint32_t minus);
Function descriptions	configure RTC smooth calibration
Precondition	-
The called functions	-
Input parameter{in}	
window	select calibration window
RTC_CALIBRATION_WINDOW_32S	2exp20 RTCCLK cycles, 32s if RTCCLK = 32768 Hz
RTC_CALIBRATION_WINDOW_16S	2exp19 RTCCLK cycles, 16s if RTCCLK = 32768 Hz
RTC_CALIBRATION_WINDOW_8S	2exp18 RTCCLK cycles, 8s if RTCCLK = 32768 Hz
Input parameter{in}	
plus	add RTC clock or not
RTC_CALIBRATION_PLUS_SET	add one RTC clock every 2048 rtc clock
RTC_CALIBRATION_PLUS_RESET	no effect
Input parameter{in}	
minus	the RTC clock to minus during the calibration window(0x0 - 0x1FF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* configure RTC smooth calibration */
```

```
ErrStatus error_status;
```

```
error_status = rtc_smooth_calibration_config(RTC_CALIBRATION_WINDOW_32S, RTC_
CALIBRATION_PLUS_SET, 0x10);
```

rtc_coarse_calibration_enable

The description of rtc_coarse_calibration_enable is shown as below:

Table 3-758. rtc_coarse_calibration_enable

Function name	rtc_coarse_calibration_enable
Function prototype	ErrStatus rtc_coarse_calibration_enable(void);
Function descriptions	enable RTC coarse calibration
Precondition	-
The called functions	rtc_init_mode_enter/rtc_init_mode_exit
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* enable RTC coarse calibration */
rtc_coarse_calibration_enable ();
```

rtc_coarse_calibration_disable

The description of rtc_coarse_calibration_disable is shown as below:

Table 3-759. rtc_coarse_calibration_disable

Function name	rtc_coarse_calibration_disable
Function prototype	ErrStatus rtc_coarse_calibration_disable(void);
Function descriptions	disable RTC coarse calibration
Precondition	-
The called functions	rtc_init_mode_enter/rtc_init_mode_exit
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* rtc_coarse_calibration_disable */
ErrStatus error_status = rtc_coarse_calibration_disable ();
```

rtc_coarse_calibration_config

The description of rtc_coarse_calibration_config is shown as below:

Table 3-760. rtc_coarse_calibration_config

Function name	rtc_coarse_calibration_config
Function prototype	ErrStatus rtc_coarse_calibration_config(uint8_t direction, uint8_t step);
Function descriptions	config coarse calibration direction and step
Precondition	-
The called functions	rtc_init_mode_enter/rtc_init_mode_exit
Input parameter{in}	
direction	coarse calibration direction
<i>CALIB_INCREASE</i>	Increase calendar update frequency
<i>CALIB_DECREASE</i>	Decrease calendar update frequency
Input parameter{in}	
step	coarse calibration step
<i>0x00-0x1F</i>	COSD=0: 0x00: +0PPM 0x01: +4PPM(approximate value) 0x02: +8PPM (approximate value) 0x1F: +126PPM (approximate value) COSD=1: 0x00: -0PPM 0x01: -2PPM(approximate value) 0x02: -4PPM (approximate value) 0x1F: -63PPM (approximate value)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* config coarse calibration direction and step */
```

```
ErrStatus error_status = rtc_coarse_calibration_config (INCREASE, 0x01);
```

rtc_pri_pro_enable

The description of rtc_pri_pro_enable is shown as below:

Table 3-761. rtc_pri_pro_enable

Function name	rtc_pri_pro_enable
Function prototype	void rtc_pri_pro_enable(uint32_t sub_area);

Function descriptions	enable RTC privilege protection mode
Precondition	-
The called functions	-
Input parameter{in}	
sub_area	specify which sub area of the RTC module to be protected by the privilege mode
<i>RTC_PPM_CTL_ALRM0PRIP</i>	Alarm 0 privilege protection
<i>RTC_PPM_CTL_ALRM1PRIP</i>	Alarm 1 privilege protection
<i>RTC_PPM_CTL_WUTPRIP</i>	Wakeup timer privilege protection
<i>RTC_PPM_CTL_TSPRIP</i>	Timestamp privilege protection
<i>RTC_PPM_CTL_TAMPPRIP</i>	Tamper privilege protection (excluding backup registers)
<i>RTC_PPM_CTL_CALCPRIP</i>	Shift register, daylight saving, calibration and reference clock privilege protection
<i>RTC_PPM_CTL_INITPRIP</i>	Initialization privilege protection
<i>RTC_PPM_CTL_RTCPRIP</i>	RTC privilege protection
<i>RTC_PPM_CTL_BKPRWPRIP</i>	backup registers zone a privilege protection
<i>RTC_PPM_CTL_BKPWPRIP</i>	backup registers zone b privilege protection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC alarm 0 privilege protection mode */
```

```
rtc_pri_pro_enable(RTC_PPM_CTL_ALRM0PRIP);
```

rtc_pri_pro_disable

The description of rtc_pri_pro_disable is shown as below:

Table 3-762. rtc_pri_pro_enable

Function name	rtc_pri_pro_disable
Function prototype	void rtc_pri_pro_disable(uint32_t sub_area);
Function descriptions	disable RTC privilege protection mode

Precondition	-
The called functions	-
Input parameter{in}	
sub_area	specify which sub area of the RTC module to be disabled protecte by the privilege mode
<i>RTC_PPM_CTL_ALRM0PRIP</i>	Alarm 0 privilege protection
<i>RTC_PPM_CTL_ALRM1PRIP</i>	Alarm 1 privilege protection
<i>RTC_PPM_CTL_WUTPRIP</i>	Wakeup timer privilege protection
<i>RTC_PPM_CTL_TSPRIP</i>	Timestamp privilege protection
<i>RTC_PPM_CTL_TAMPPRIP</i>	Tamper privilege protection (excluding backup registers)
<i>RTC_PPM_CTL_CALCPRIP</i>	Shift register, daylight saving, calibration and reference clock privilege protection
<i>RTC_PPM_CTL_INITPRIP</i>	Initialization privilege protection
<i>RTC_PPM_CTL_RTCPRIP</i>	RTC privilege protection
<i>RTC_PPM_CTL_BKPRWPRIP</i>	backup registers zone a privilege protection
<i>RTC_PPM_CTL_BKPWPRIP</i>	backup registers zone b privilege protection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC alarm 0 privilege protection mode */
```

```
rtc_pri_pro_disable(RTC_PPM_CTL_ALRM0PRIP);
```

rtc_sec_pro_enable

The description of rtc_sec_pro_enable is shown as below:

Table 3-763. rtc_sec_pro_enable

Function name	rtc_sec_pro_enable
Function prototype	void rtc_sec_pro_enable(uint32_t sub_area);
Function descriptions	enable RTC secure protection mode
Precondition	-

The called functions	-
Input parameter{in}	
sub_area	specify which sub area of the RTC module to be protected by the secure mode
<i>RTC_SPM_CTL_ALRM0SECP</i>	Alarm 0 secure protection
<i>RTC_SPM_CTL_ALRM1SECP</i>	Alarm 1 secure protection
<i>RTC_SPM_CTL_WUTSECP</i>	Wakeup timer secure protection
<i>RTC_SPM_CTL_TSSECP</i>	Timestamp secure protection
<i>RTC_SPM_CTL_TAMPSECP</i>	Tamper secure protection (excluding backup registers)
<i>RTC_SPM_CTL_CALCSECP</i>	Shift register, daylight saving, calibration and reference clock secure protection
<i>RTC_SPM_CTL_INITSECP</i>	Initialization secure protection
<i>RTC_SPM_CTL_RTCSSECP</i>	RTC global secure protection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC alarm 0 secure protection mode */
```

```
rtc_sec_pro_enable (RTC_SPM_CTL_ALRM0SECP);
```

rtc_sec_pro_disable

The description of rtc_sec_pro_disable is shown as below:

Table 3-764. rtc_sec_pro_disable

Function name	rtc_sec_pro_disable
Function prototype	void rtc_sec_pro_disable(uint32_t sub_area);
Function descriptions	disable RTC secure protection mode
Precondition	-
The called functions	-
Input parameter{in}	
sub_area	specify which sub area of the RTC module to be disabled protecte by the secure mode
<i>RTC_SPM_CTL_ALRM</i>	Alarm 0 secure protection

<i>0SECP</i>	
<i>RTC_SPM_CTL_ALRM1SECP</i>	Alarm 1 secure protection
<i>RTC_SPM_CTL_WUTSECP</i>	Wakeup timer secure protection
<i>RTC_SPM_CTL_TSSECP</i>	Timestamp secure protection
<i>RTC_SPM_CTL_TAMPSECP</i>	Tamper secure protection (excluding backup registers)
<i>RTC_SPM_CTL_CALCSECP</i>	Shift register, daylight saving, calibration and reference clock secure protection
<i>RTC_SPM_CTL_INITSECP</i>	Initialization secure protection
<i>RTC_SPM_CTL_RTCSSECP</i>	RTC global secure protection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*disable RTC alarm 0 secure protection mode */
```

```
rtc_sec_pro_disable (RTC_SPM_CTL_ALRM0SECP);
```

rtc_bkp_zone_a_sec_pro_set

The description of rtc_bkp_zone_a_sec_pro_set is shown as below:

Table 3-765. rtc_bkp_zone_a_sec_pro_set

Function name	rtc_bkp_zone_a_sec_pro_set
Function prototype	void rtc_bkp_zone_a_sec_pro_set(uint32_t zone_a_tail);
Function descriptions	set the RTC_BKP secure protection zone_a tail value
Precondition	-
The called functions	-
Input parameter{in}	
zone_a_tail	RTC_BKP registers secure protection zone_a
<i>RTC_BKP_PRO_ZONE_A_TAIL_NONE</i>	there is no RTC_BKP registers secure protection zone_a
<i>RTC_BKP_PRO_ZONE_A_TAIL_x</i>	RTC_BKP registers secure protection zone_a is from RTC_BKP0 to RTC_BKPx, which can be read and written only when the APB is in secure mode (x = 0,1,2,...19)
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* set the RTC_BKP secure protection zonea tail value */
rtc_bkp_zoneb_sec_pro_set (RTC_BKP_PRO_ZONEA_TAIL_3);
```

rtc_bkp_zoneb_sec_pro_set

The description of rtc_bkp_zoneb_sec_pro_set is shown as below:

Table 3-766. rtc_bkp_zoneb_sec_pro_set

Function name	rtc_bkp_zoneb_sec_pro_set
Function prototype	ErrStatus rtc_bkp_zoneb_sec_pro_set(uint32_t zoneb_tail);
Function descriptions	set the RTC_BKP secure protection zoneb tail value
Precondition	-
The called functions	-
Input parameter{in}	
zoneb_tail	RTC_BKP registers secure protection zoneb
RTC_BKP_PRO_ZONEB_TAIL_NONE	there is no RTC_BKP registers secure protection zoneb
RTC_BKP_PRO_ZONEB_TAIL_x	RTC_BKP registers secure protection zonea is from RTC_BKP0 to RTC_BKPx, which can be read and written only when the APB is in secure mode (x = 0,1,2,...19)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* set the RTC_BKP secure protection zoneb tail value */
ErrStatus error_states = rtc_bkp_zoneb_sec_pro_set(RTC_BKP_PRO_ZONEB_TAIL_3);
```

rtc_bkp_zoneb_sec_pro_check

The description of rtc_bkp_zoneb_sec_pro_check is shown as below:

Table 3-767. rtc_bkp_zoneb_sec_pro_check

Function name	rtc_bkp_zoneb_sec_pro_check
Function prototype	ErrStatus rtc_bkp_zoneb_sec_pro_check(void);
Function descriptions	check the RTC_BKP secure protection zoneb is valid or not
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* check the RTC_BKP secure protection zoneb is valid or not */
```

```
ErrStatus error_states = rtc_bkp_zoneb_sec_pro_check ();
```

3.23. SDIO

The secure digital input/output interface (SDIO) defines the SD/SD I/O /MMC CE-ATA card host interface, which provides command/data transfer between the AHB system bus and SD memory cards, SD I/O cards, Multimedia Card (MMC), and CE-ATA devices. The SDIO registers are listed in chapter [3.23.1](#), the SDIO firmware functions are introduced in chapter [3.23.2](#).

3.23.1. Descriptions of Peripheral registers

SDIO registers are listed in the table shown as below:

Table 3-768. SDIO Registers

Registers	Descriptions
SDIO_PWRCTL	Power control register
SDIO_CLKCTL	Clock control register
SDIO_CMDAGMT	Command argument register
SDIO_CMDCTL	Command control register
SDIO_RSPCMDIDX	Command index response register
SDIO_RESPx x=0..3	Response register
SDIO_DATATO	Data timeout register
SDIO_DATALEN	Data length register
SDIO_DATACTL	Data control register
SDIO_DATACNT	Data counter register
SDIO_STAT	Status register
SDIO_INTC	Interrupt clear register
SDIO_INTEN	Interrupt enable register
SDIO_FIFOCNT	FIFO counter register
SDIO_FIFO	FIFO data register

3.23.2. Descriptions of Peripheral functions

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

Table 3-769. SDIO firmware function

Function name	Function description
sdio_deinit	deinitialize the SDIO
sdio_clock_config	configure the SDIO clock
sdio_hardware_clock_enable	enable hardware clock control
sdio_hardware_clock_disable	disable hardware clock control
sdio_bus_mode_set	set different SDIO card bus mode
sdio_power_state_set	set the SDIO power state
sdio_power_state_get	get the SDIO power state
sdio_clock_enable	enable SDIO_CLK clock output
sdio_clock_disable	disable SDIO_CLK clock output
sdio_command_response_config	configure the command and response
sdio_wait_type_set	set the command state machine wait type
sdio_csm_enable	enable the CSM(command state machine)
sdio_csm_disable	disable the CSM(command state machine)
sdio_command_index_get	get the last response command index
sdio_response_get	get the response for the last received command
sdio_data_config	configure the data timeout, data length and data block size
sdio_data_transfer_config	configure the data transfer mode and direction
sdio_dsm_enable	enable the DSM(data state machine) for data transfer
sdio_dsm_disable	disable the DSM(data state machine)
sdio_data_write	write data(one word) to the transmit FIFO
sdio_data_read	read data(one word) from the receive FIFO
sdio_data_counter_get	get the number of remaining data bytes to be transferred to card
sdio_fifo_counter_get	get the number of words remaining to be written or read from FIFO
sdio_dma_enable	enable the DMA request for SDIO
sdio_dma_disable	disable the DMA request for SDIO
sdio_flag_get	get the flags state of SDIO
sdio_flag_clear	clear the pending flags of SDIO
sdio_interrupt_enable	enable the SDIO interrupt
sdio_interrupt_disable	disable the SDIO interrupt
sdio_interrupt_flag_get	get the interrupt flags state of SDIO
sdio_interrupt_flag_clear	clear the interrupt pending flags of SDIO
sdio_readwait_enable	enable the read wait mode(SD I/O only)
sdio_readwait_disable	disable the read wait mode(SD I/O only)
sdio_stop_readwait_enable	enable the function that stop the read wait process(SD I/O only)

Function name	Function description
sdio_stop_readwait_disable	disable the function that stop the read wait process(SD I/O only)
sdio_readwait_type_set	set the read wait type(SD I/O only)
sdio_operation_enable	enable the SD I/O mode specific operation(SD I/O only)
sdio_operation_disable	disable the SD I/O mode specific operation(SD I/O only)
sdio_suspend_enable	enable the SD I/O suspend operation(SD I/O only)
sdio_suspend_disable	disable the SD I/O suspend operation(SD I/O only)
sdio_ceata_command_enable	enable the CE-ATA command(CE-ATA only)
sdio_ceata_command_disable	disable the CE-ATA command(CE-ATA only)
sdio_ceata_interrupt_enable	enable the CE-ATA interrupt(CE-ATA only)
sdio_ceata_interrupt_disable	disable the CE-ATA interrupt(CE-ATA only)
sdio_ceata_command_completion_enable	enable the CE-ATA command completion signal(CE-ATA only)
sdio_ceata_command_completion_disable	disable the CE-ATA command completion signal(CE-ATA only)

sdio_deinit

The description of sdio_deinit is shown as below:

Table 3-770. Function sdio_deinit

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

Example:

```
/* deinitialize the SDIO */
sdio_deinit();
```

sdio_clock_config

The description of sdio_clock_config is shown as below:

Table 3-771. Function sdio_clock_config

Function name	sdio_clock_config
---------------	-------------------

Function prototype	void sdio_clock_config(uint32_t clock_edge, uint32_t clock_bypass, uint32_t clock_powersave, uint16_t clock_division);
Function descriptions	configure the SDIO clock
Precondition	-
The called functions	-
Input parameter{in}	
clock_edge	SDIO_CLK clock edge
SDIO_SDIOLCKEDGE_RISING	select the rising edge of the SDIOCLK to generate SDIO_CLK
SDIO_SDIOLCKEDGE_FALLING	select the falling edge of the SDIOCLK to generate SDIO_CLK
Input parameter{in}	
clock_bypass	clock bypass
SDIO_CLOCKBYPASS_ENABLE	clock bypass
SDIO_CLOCKBYPASS_DISABLE	no bypass
Input parameter{in}	
clock_powersave	SDIO_CLK clock dynamic switch on/off for power saving
SDIO_CLOCKPWRSAVE_ENABLE	SDIO_CLK closed when bus is idle
SDIO_CLOCKPWRSAVE_DISABLE	SDIO_CLK clock is always on
Input parameter{in}	
clock_division	clock division, less than 512
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the SDIO clock */
```

```
sdio_clock_config(SDIO_SDIOLCKEDGE_RISING, SDIO_CLOCKBYPASS_DISABLE,
SDIO_CLOCKPWRSAVE_DISABLE, SD_CLK_DIV_TRANS);
```

sdio_hardware_clock_enable

The description of sdio_hardware_clock_enable is shown as below:

Table 3-772. Function sdio_hardware_clock_enable

Function name	sdio_hardware_clock_enable
Function prototype	void sdio_hardware_clock_enable(void);
Function descriptions	enable hardware clock control

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

Example:

```
/* enable hardware clock control */
```

```
sdio_hardware_clock_enable();
```

sdio_hardware_clock_disable

The description of sdio_hardware_clock_disable is shown as below:

Table 3-773. Function sdio_hardware_clock_disable

Function name	sdio_hardware_clock_disable
Function prototype	void sdio_hardware_clock_disable(void);
Function descriptions	disable hardware clock control
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable hardware clock control */
```

```
sdio_hardware_clock_disable();
```

sdio_bus_mode_set

The description of sdio_bus_mode_set is shown as below:

Table 3-774. Function sdio_bus_mode_set

Function name	sdio_bus_mode_set
Function prototype	void sdio_bus_mode_set(uint32_t bus_mode);
Function descriptions	set different SDIO card bus mode
Precondition	-
The called functions	-

Input parameter{in}	
bus_mode	SDIO card bus mode
<i>SDIO_BUSMODE_1BIT</i> <i>T</i>	1-bit SDIO card bus mode
<i>SDIO_BUSMODE_4BIT</i> <i>T</i>	4-bit SDIO card bus mode
<i>SDIO_BUSMODE_8BIT</i> <i>T</i>	8-bit SDIO card bus mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO bus mode */
sdio_bus_mode_set(SDIO_BUSMODE_1BIT);
```

sdio_power_state_set

The description of sdio_power_state_set is shown as below:

Table 3-775. Function sdio_power_state_set

Function name	sdio_power_state_set
Function prototype	void sdio_power_state_set(uint32_t power_state);
Function descriptions	set the SDIO power state
Precondition	-
The called functions	-
Input parameter{in}	
power_state	SDIO power state
<i>SDIO_POWER_ON</i>	SDIO power on
<i>SDIO_POWER_OFF</i>	SDIO power off
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SDIO power state */
sdio_power_state_set(SDIO_POWER_ON);
```

sdio_power_state_get

The description of sdio_power_state_get is shown as below:

Table 3-776. Function `sdio_power_state_get`

Function name	<code>sdio_power_state_get</code>
Function prototype	<code>uint32_t sdio_power_state_get(void);</code>
Function descriptions	get the SDIO power state
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>uint32_t</code>	SDIO_POWER_ON / SDIO_POWER_OFF

Example:

```
/* get the SDIO power state */
uint32_t sdio_power_value;
sdio_power_value = sdio_power_state_get();
```

sdio_clock_enable

The description of `sdio_clock_enable` is shown as below:

Table 3-777. Function `sdio_clock_enable`

Function name	<code>sdio_clock_enable</code>
Function prototype	<code>void sdio_clock_enable(void);</code>
Function descriptions	enable SDIO_CLK clock output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SDIO_CLK clock output */
sdio_clock_enable();
```

sdio_clock_disable

The description of `sdio_clock_disable` is shown as below:

Table 3-778. Function `sdio_clock_disable`

Function name	<code>sdio_clock_disable</code>
Function prototype	<code>void sdio_clock_disable(void);</code>
Function descriptions	disable SDIO_CLK clock output
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SDIO_CLK clock output */
sdio_clock_disable();
```

`sdio_command_response_config`

The description of `sdio_command_response_config` is shown as below:

Table 3-779. Function `sdio_command_response_config`

Function name	<code>sdio_command_response_config</code>
Function prototype	<code>void sdio_command_response_config(uint32_t cmd_index, uint32_t cmd_argument, uint32_t response_type);</code>
Function descriptions	configure the command and response
Precondition	-
The called functions	-
Input parameter{in}	
<code>cmd_index</code>	command index, refer to the related specifications
Input parameter{in}	
<code>cmd_argument</code>	command argument, refer to the related specifications
Input parameter{in}	
<code>response_type</code>	response type
<code>SDIO_RESPONSETYPE_NO</code>	no response
<code>SDIO_RESPONSETYPE_SHORT</code>	short response
<code>SDIO_RESPONSETYPE_LONG</code>	long response
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* CMD2(SD_CMD_ALL_SEND_CID) command response config*/

sdio_command_response_config(SD_CMD_ALL_SEND_CID, (uint32_t)0x0,
SDIO_RESPONSETYPE_LONG);
```

sdio_wait_type_set

The description of sdio_wait_type_set is shown as below:

Table 3-780. Function sdio_wait_type_set

Function name	sdio_wait_type_set
Function prototype	void sdio_wait_type_set(uint32_t wait_type);
Function descriptions	set the command state machine wait type
Precondition	-
The called functions	-
Input parameter{in}	
wait_type	wait type
SDIO_WAITTYPE_NO	not wait interrupt
SDIO_WAITTYPE_INTERRUPT	wait interrupt
SDIO_WAITTYPE_DATAEND	wait the end of data transfer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the command state machine wait type */

sdio_wait_type_set(SDIO_WAITTYPE_NO);
```

sdio_csm_enable

The description of sdio_csm_enable is shown as below:

Table 3-781. Function sdio_csm_enable

Function name	sdio_csm_enable
Function prototype	void sdio_csm_enable(void);
Function descriptions	enable the CSM(command state machine)
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* enable the CSM(command state machine) */
```

```
sdio_csm_enable();
```

sdio_csm_disable

The description of sdio_csm_disable is shown as below:

Table 3-782. Function sdio_csm_disable

Function name	sdio_csm_disable
Function prototype	void sdio_csm_disable(void);
Function descriptions	disable the CSM(command state machine)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CSM(command state machine) */
```

```
sdio_csm_disable();
```

sdio_command_index_get

The description of sdio_command_index_get is shown as below:

Table 3-783. Function sdio_command_index_get

Function name	sdio_command_index_get
Function prototype	uint8_t sdio_command_index_get(void);
Function descriptions	get the last response command index
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
uint8_t	last response command index

Example:

```
/* get SDIO command index */

uint8_t sdio_commond_value;

sdio_commond_value = sdio_command_index_get();
```

sdio_response_get

The description of sdio_response_get is shown as below:

Table 3-784. Function sdio_response_get

Function name	sdio_response_get
Function prototype	uint32_t sdio_response_get(uint32_t responsex);
Function descriptions	get the response for the last received command
Precondition	-
The called functions	-
Input parameter{in}	
responsex	SDIO response
<i>SDIO_RESPONSE0</i>	card response[31:0]/card response[127:96]
<i>SDIO_RESPONSE1</i>	card response[95:64]
<i>SDIO_RESPONSE2</i>	card response[63:32]
<i>SDIO_RESPONSE3</i>	card response[31:1], plus bit 0
Output parameter{out}	
-	-
Return value	
uint32_t	response for the last received command

Example:

```
/* store the CID0 numbers */

uint32_t sdio_cid[4] = {0, 0, 0, 0};

sdio_cid[0] = sdio_response_get(SDIO_RESPONSE0);
```

sdio_data_config

The description of sdio_data_config is shown as below:

Table 3-785. Function sdio_data_config

Function name	sdio_data_config
Function prototype	void sdio_data_config(uint32_t data_timeout, uint32_t data_length, uint32_t data_blocksize);

Function descriptions	configure the data timeout, data length and data block size
Precondition	-
The called functions	-
Input parameter{in}	
data_timeout	data timeout period in card bus clock periods
Input parameter{in}	
data_length	number of data bytes to be transferred
Input parameter{in}	
data_blocksize	size of data block for block transfer
SDIO_DATABLOCKSIZE E_1BYTE	block size = 1 byte
SDIO_DATABLOCKSIZE E_2BYTES	block size = 2 bytes
SDIO_DATABLOCKSIZE E_4BYTES	block size = 4 bytes
SDIO_DATABLOCKSIZE E_8BYTES	block size = 8 bytes
SDIO_DATABLOCKSIZE E_16BYTES	block size = 16 bytes
SDIO_DATABLOCKSIZE E_32BYTES	block size = 32 bytes
SDIO_DATABLOCKSIZE E_64BYTES	block size = 64 bytes
SDIO_DATABLOCKSIZE E_128BYTES	block size = 128 bytes
SDIO_DATABLOCKSIZE E_256BYTES	block size = 256 bytes
SDIO_DATABLOCKSIZE E_512BYTES	block size = 512 bytes
SDIO_DATABLOCKSIZE E_1024BYTES	block size = 1024 bytes
SDIO_DATABLOCKSIZE E_2048BYTES	block size = 2048 bytes
SDIO_DATABLOCKSIZE E_4096BYTES	block size = 4096 bytes
SDIO_DATABLOCKSIZE E_8192BYTES	block size = 8192 bytes
SDIO_DATABLOCKSIZE E_16384BYTES	block size = 16384 bytes
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SDIO data */

sdio_data_config(0, 0, SDIO_DATABLOCKSIZE_1BYTE);
```

sdio_data_transfer_config

The description of sdio_data_transfer_config is shown as below:

Table 3-786. Function sdio_data_transfer_config

Function name	sdio_data_transfer_config
Function prototype	void sdio_data_transfer_config(uint32_t transfer_mode, uint32_t transfer_direction);
Function descriptions	configure the data transfer mode and direction
Precondition	-
The called functions	-
Input parameter{in}	
transfer_mode	mode of data transfer
SDIO_TRANSMODE_BLOCK	block transfer
SDIO_TRANSMODE_STREAM	stream transfer or SDIO multibyte transfer
Input parameter{in}	
transfer_direction	data transfer direction, read or write
SDIO_TRANSDIRECTION_TOCARD	write data to card
SDIO_TRANSDIRECTION_TOSDIO	read data from card
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SDIO data transmisson */

sdio_data_transfer_config(SDIO_TRANSDIRECTION_TOSDIO,
SDIO_TRANSMODE_BLOCK);
```

sdio_dsm_enable

The description of sdio_dsm_enable is shown as below:

Table 3-787. Function sdio_dsm_enable

Function name	sdio_dsm_enable
----------------------	-----------------

Function prototype	void sdio_dsm_enable(void);
Function descriptions	enable the DSM(data state machine) for data transfer
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the DSM(data state machine) */
sdio_dsm_enable();
```

sdio_dsm_disable

The description of sdio_dsm_disable is shown as below:

Table 3-788. Function sdio_dsm_disable

Function name	sdio_dsm_disable
Function prototype	void sdio_dsm_disable(void);
Function descriptions	disable the DSM(data state machine)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the DSM(data state machine) */
sdio_dsm_disable();
```

sdio_data_write

The description of sdio_data_write is shown as below:

Table 3-789. Function sdio_data_write

Function name	sdio_data_write
Function prototype	void sdio_data_write(uint32_t data);
Function descriptions	write data(one word) to the transmit FIFO

Precondition	-
The called functions	-
Input parameter{in}	
data	32-bit data write to card
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write data(one word) to the transmit FIFO */
```

```
sdio_data_write(0x0000 0001);
```

sdio_data_read

The description of sdio_data_read is shown as below:

Table 3-790. Function sdio_data_read

Function name	sdio_data_read
Function prototype	uint32_t sdio_data_read(void);
Function descriptions	read data(one word) from the receive FIFO
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	received data

Example:

```
/* read data(one word) from the receive FIFO */
```

```
sdio_data_read();
```

sdio_data_counter_get

The description of sdio_data_counter_get is shown as below:

Table 3-791. Function sdio_data_counter_get

Function name	sdio_data_counter_get
Function prototype	uint32_t sdio_data_counter_get(void);
Function descriptions	get the number of remaining data bytes to be transferred to card
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	number of remaining data bytes to be transferred

Example:

```
/* get the number of remaining data bytes to be transferred to card */
```

```
uint32_t sdio_data_value;
```

```
sdio_data_value = sdio_data_counter_get();
```

sdio_fifo_counter_get

The description of sdio_fifo_counter_get is shown as below:

Table 3-792. Function sdio_data_counter_get

Function name	sdio_fifo_counter_get
Function prototype	uint32_t sdio_fifo_counter_get(void);
Function descriptions	get the number of words remaining to be written or read from FIFO
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	remaining number of words

Example:

```
/* get the number of words remaining to be written or read from FIFO */
```

```
uint32_t sdio_fifo_value;
```

```
sdio_fifo_value = sdio_fifo_counter_get();
```

sdio_dma_enable

The description of sdio_dma_enable is shown as below:

Table 3-793. Function sdio_dma_enable

Function name	sdio_dma_enable
Function prototype	void sdio_dma_enable(void);
Function descriptions	enable the DMA request for SDIO
Precondition	-

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

Example:

```
/* enable the SDIO DMA */
```

```
sdio_dma_enable();
```

sdio_dma_disable

The description of sdio_dma_disable is shown as below:

Table 3-794. Function sdio_dma_disable

Function name	sdio_dma_disable
Function prototype	void sdio_dma_disable(void);
Function descriptions	disable the DMA request for SDIO
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the SDIO DMA */
```

```
sdio_dma_disable();
```

sdio_flag_get

The description of sdio_flag_get is shown as below:

Table 3-795. Function sdio_flag_get

Function name	sdio_flag_get
Function prototype	FlagStatus sdio_flag_get(uint32_t flag);
Function descriptions	get the flags state of SDIO
Precondition	-
The called functions	-
Input parameter{in}	

flag	flags state of SDIO
<i>SDIO_FLAG_CCR CER R</i>	command response received (CRC check failed) flag
<i>SDIO_FLAG_DTCRCE RR</i>	data block sent/received (CRC check failed) flag
<i>SDIO_FLAG_CMDTMO UT</i>	command response timeout flag
<i>SDIO_FLAG_DTTMOU T</i>	data timeout flag
<i>SDIO_FLAG_TXURE</i>	transmit FIFO underrun error occurs flag
<i>SDIO_FLAG_RXORE</i>	received FIFO overrun error occurs flag
<i>SDIO_FLAG_CMDREC V</i>	command response received (CRC check passed) flag
<i>SDIO_FLAG_CMDSEN D</i>	command sent (no response required) flag
<i>SDIO_FLAG_DTEND</i>	data end (data counter, SDIO_DATA_CNT, is zero) flag
<i>SDIO_FLAG_STBITE</i>	start bit error in the bus flag
<i>SDIO_FLAG_DTBLKE ND</i>	data block sent/received (CRC check passed) flag
<i>SDIO_FLAG_CMDRUN</i>	command transmission in progress flag
<i>SDIO_FLAG_TXRUN</i>	data transmission in progress flag
<i>SDIO_FLAG_RXRUN</i>	data reception in progress flag
<i>SDIO_FLAG_TFH</i>	transmit FIFO is half empty flag: at least 8 words can be written into the FIFO
<i>SDIO_FLAG_RFH</i>	receive FIFO is half full flag: at least 8 words can be read in the FIFO
<i>SDIO_FLAG_TFF</i>	transmit FIFO is full flag
<i>SDIO_FLAG_RFF</i>	receive FIFO is full flag
<i>SDIO_FLAG_TFE</i>	transmit FIFO is empty flag
<i>SDIO_FLAG_RFE</i>	receive FIFO is empty flag
<i>SDIO_FLAG_TXDTVAL</i>	data is valid in transmit FIFO flag
<i>SDIO_FLAG_RXDTVA L</i>	data is valid in receive FIFO flag
<i>SDIO_FLAG_SDIOINT</i>	SD I/O interrupt received flag
<i>SDIO_FLAG_ATAEND</i>	CE-ATA command completion signal received (only for CMD61) flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the flags state of SDIO */
```

```
FlagStatus flag_value;
```

```
flag_value = sdio_flag_get(SDIO_FLAG_RFH);
```

sdio_flag_clear

The description of sdio_flag_clear is shown as below:

Table 3-796. Function sdio_flag_clear

Function name	sdio_flag_clear
Function prototype	void sdio_flag_clear(uint32_t flag);
Function descriptions	clear the pending flags of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
flag	flags state of SDIO
SDIO_FLAG_CCR CER R	command response received (CRC check failed) flag
SDIO_FLAG_DTC RC E RR	data block sent/received (CRC check failed) flag
SDIO_FLAG_CMDTMO UT	command response timeout flag
SDIO_FLAG_DTTMOU T	data timeout flag
SDIO_FLAG_TXURE	transmit FIFO underrun error occurs flag
SDIO_FLAG_RXORE	received FIFO overrun error occurs flag
SDIO_FLAG_CMDREC V	command response received (CRC check passed) flag
SDIO_FLAG_CMDSEN D	command sent (no response required) flag
SDIO_FLAG_DTEND	data end (data counter, SDIO_DATA_CNT, is zero) flag
SDIO_FLAG_STBITE	start bit error in the bus flag
SDIO_FLAG_DTB LKE ND	data block sent/received (CRC check passed) flag
SDIO_FLAG_SDIOINT	SD I/O interrupt received flag
SDIO_FLAG_ATAEND	CE-ATA command completion signal received (only for CMD61) flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the pending flags of SDIO */
```

```
sdio_flag_clear(SDIO_FLAG_DTCRCERR);
```

sdio_interrupt_enable

The description of sdio_interrupt_enable is shown as below:

Table 3-797. Function sdio_interrupt_enable

Function name	sdio_interrupt_enable
Function prototype	void sdio_interrupt_enable(uint32_t int_flag);
Function descriptions	enable the SDIO interrupt
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt flags state of SDIO
SDIO_INT_CCRERR	SDIO CCRERR interrupt
SDIO_INT_DTCRCERR	SDIO DTCRCERR interrupt
SDIO_INT_CMDTMOUT	SDIO CMDTMOUT interrupt
SDIO_INT_DTTMOUT	SDIO DTTMOUT interrupt
SDIO_INT_TXURE	SDIO TXURE interrupt
SDIO_INT_RXORE	SDIO_INT_RXORE
SDIO_INT_CMDRECV	SDIO CMDRECV interrupt
SDIO_INT_CMDSEND	SDIO CMDSEND interrupt
SDIO_INT_DTEND	SDIO DTEND interrupt
SDIO_INT_STBITE	SDIO STBITE interrupt
SDIO_INT_DTBLKEND	SDIO DTBLKEND interrupt
SDIO_INT_CMDRUN	SDIO CMDRUN interrupt
SDIO_INT_TXRUN	SDIO TXRUN interrupt
SDIO_INT_RXRUN	SDIO RXRUN interrupt
SDIO_INT_TFH	SDIO TFH interrupt
SDIO_INT_RFH	SDIO RFH interrupt
SDIO_INT_TFF	SDIO TFF interrupt
SDIO_INT_RFF	SDIO RFF interrupt
SDIO_INT_TFE	SDIO TFE interrupt
SDIO_INT_RFE	SDIO RFE interrupt
SDIO_INT_TXDTVAL	SDIO TXDTVAL interrupt
SDIO_INT_RXDTVAL	SDIO RXDTVAL interrupt
SDIO_INT_SDIOINT	SDIO SDIOINT interrupt
SDIO_INT_ATAEND	SDIO ATAEND interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

/* enable the SDIO corresponding interrupts */

```
sdio_interrupt_enable(SDIO_INT_CCRRCERR | SDIO_INT_DTTMOUT | SDIO_INT_RXORE
| SDIO_INT_DTEND | SDIO_INT_STBITE);
```

sdio_interrupt_disable

The description of sdio_interrupt_disable is shown as below:

Table 3-798. Function sdio_interrupt_disable

Function name	sdio_interrupt_disable
Function prototype	void sdio_interrupt_disable(uint32_t int_flag);
Function descriptions	disable the SDIO interrupt
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt flags state of SDIO
SDIO_INT_CCRRCERR	SDIO CCRRCERR interrupt
SDIO_INT_DTCRCERR	SDIO DTCRCERR interrupt
SDIO_INT_CMDTMOUT	SDIO CMDTMOUT interrupt
SDIO_INT_DTTMOUT	SDIO DTTMOUT interrupt
SDIO_INT_TXURE	SDIO TXURE interrupt
SDIO_INT_RXORE	SDIO_INT_RXORE
SDIO_INT_CMDRECV	SDIO CMDRECV interrupt
SDIO_INT_CMDSEND	SDIO CMDSEND interrupt
SDIO_INT_DTEND	SDIO DTEND interrupt
SDIO_INT_STBITE	SDIO STBITE interrupt
SDIO_INT_DTBLKEND	SDIO DTBLKEND interrupt
SDIO_INT_CMDRUN	SDIO CMDRUN interrupt
SDIO_INT_TXRUN	SDIO TXRUN interrupt
SDIO_INT_RXRUN	SDIO RXRUN interrupt
SDIO_INT_TFH	SDIO TFH interrupt
SDIO_INT_RFH	SDIO RFH interrupt
SDIO_INT_TFF	SDIO TFF interrupt
SDIO_INT_RFF	SDIO RFF interrupt
SDIO_INT_TFE	SDIO TFE interrupt
SDIO_INT_RFE	SDIO RFE interrupt
SDIO_INT_TXDTVAL	SDIO TXDTVAL interrupt
SDIO_INT_RXDTVAL	SDIO RXDTVAL interrupt
SDIO_INT_SDIOINT	SDIO SDIOINT interrupt
SDIO_INT_ATAEND	SDIO ATAEND interrupt
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable the SDIO interrupt */
```

```
sdio_interrupt_disable(SDIO_INT_DTCRCERR);
```

sdio_interrupt_flag_get

The description of sdio_interrupt_flag_get is shown as below:

Table 3-799. Function sdio_interrupt_flag_get

Function name	sdio_interrupt_flag_get
Function prototype	FlagStatus sdio_interrupt_flag_get(uint32_t int_flag);
Function descriptions	get the interrupt flags state of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt flags state of SDIO
SDIO_INT_FLAG_CCR CERR	SDIO CCRCERR interrupt flag
SDIO_INT_FLAG_DTC RCERR	SDIO DTCRCERR interrupt flag
SDIO_INT_FLAG_CMD TMOUT	SDIO CMDTMOUT interrupt flag
SDIO_INT_FLAG_DTT MOUT	SDIO DTTMOUT interrupt flag
SDIO_INT_FLAG_TXU RE	SDIO TXURE interrupt flag
SDIO_INT_FLAG_RXO RE	SDIO_INT_RXORE flag
SDIO_INT_FLAG_CMD RECV	SDIO CMDRECV interrupt flag
SDIO_INT_FLAG_CMD SEND	SDIO CMDSEND interrupt flag
SDIO_INT_FLAG_DTE ND	SDIO DTEND interrupt flag
SDIO_INT_FLAG_STBI TE	SDIO STBITE interrupt flag
SDIO_INT_FLAG_DTB LKEND	SDIO DTBLKEND interrupt flag
SDIO_INT_FLAG_CMD	SDIO CMDRUN interrupt flag

<i>RUN</i>	
<i>SDIO_INT_FLAG_TXRUN</i>	SDIO TXRUN interrupt flag
<i>SDIO_INT_FLAG_RXRUN</i>	SDIO RXRUN interrupt flag
<i>SDIO_INT_FLAG_TFH</i>	SDIO TFH interrupt flag
<i>SDIO_INT_FLAG_RFH</i>	SDIO RFH interrupt flag
<i>SDIO_INT_FLAG_TFF</i>	SDIO TFF interrupt flag
<i>SDIO_INT_FLAG_RFF</i>	SDIO RFF interrupt flag
<i>SDIO_INT_FLAG_TFE</i>	SDIO TFE interrupt flag
<i>SDIO_INT_FLAG_RFE</i>	SDIO RFE interrupt flag
<i>SDIO_INT_FLAG_TXDTVAL</i>	SDIO TXDTVAL interrupt flag
<i>SDIO_INT_FLAG_RXDTVAL</i>	SDIO RXDTVAL interrupt flag
<i>SDIO_INT_FLAG_SDI</i>	SDIO SDIOINT interrupt flag
<i>SDIO_INT_FLAG_ATAEND</i>	SDIO ATAEND interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the interrupt flags state of SDIO */
```

```
FlagStatus flag_value;
```

```
flag_value = sdio_interrupt_flag_get(SDIO_INT_FLAG_DTEND);
```

sdio_interrupt_flag_clear

The description of sdio_interrupt_flag_clear is shown as below:

Table 3-800. Function sdio_interrupt_flag_clear

Function name	sdio_interrupt_flag_clear
Function prototype	void sdio_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear the interrupt pending flags of SDIO
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	interrupt flags state of SDIO
<i>SDIO_INT_FLAG_CCR</i> <i>CERR</i>	command response received (CRC check failed) flag

<i>SDIO_INT_FLAG_DTC RCERR</i>	data block sent/received (CRC check failed) flag
<i>SDIO_INT_FLAG_CMD TMOUT</i>	command response timeout flag
<i>SDIO_INT_FLAG_DTT MOUT</i>	data timeout flag
<i>SDIO_INT_FLAG_TXU RE</i>	transmit FIFO underrun error occurs flag
<i>SDIO_INT_FLAG_RXO RE</i>	received FIFO overrun error occurs flag
<i>SDIO_INT_FLAG_CMD RECV</i>	command response received (CRC check passed) flag
<i>SDIO_INT_FLAG_CMD SEND</i>	command sent (no response required) flag
<i>SDIO_INT_FLAG_DTE ND</i>	data end (data counter, SDIO_DATACNT, is zero) flag
<i>SDIO_INT_FLAG_STBI TE</i>	start bit error in the bus flag
<i>SDIO_INT_FLAG_DTB LKEND</i>	data block sent/received (CRC check passed) flag
<i>SDIO_INT_FLAG_SDI OINT</i>	SD I/O interrupt received flag
<i>SDIO_INT_FLAG_ATA END</i>	CE-ATA command completion signal received (only for CMD61) flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the interrupt pending flags of SDIO */
```

```
sdio_interrupt_flag_clear(SDIO_INT_FLAG_DTEND);
```

sdio_readwait_enable

The description of sdio_readwait_enable is shown as below:

Table 3-801. Function sdio_readwait_enable

Function name	sdio_readwait_enable
Function prototype	void sdio_readwait_enable(void);
Function descriptions	enable the read wait mode(SD I/O only)
Precondition	-
The called functions	-

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

Example:

```
/* enable the read wait mode(SD I/O only) */
```

```
sdio_readwait_enable();
```

sdio_readwait_disable

The description of sdio_readwait_disable is shown as below:

Table 3-802. Function sdio_readwait_disable

Function name	sdio_readwait_disable
Function prototype	void sdio_readwait_disable(void);
Function descriptions	disable the read wait mode(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the read wait mode(SD I/O only) */
```

```
sdio_readwait_disable();
```

sdio_stop_readwait_enable

The description of sdio_stop_readwait_enable is shown as below:

Table 3-803. Function sdio_stop_readwait_enable

Function name	sdio_stop_readwait_enable
Function prototype	void sdio_stop_readwait_enable(void);
Function descriptions	enable the function that stop the read wait process(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the function that stop the read wait process(SD I/O only) */
sdio_stop_readwait_enable();
```

sdio_stop_readwait_disable

The description of sdio_stop_readwait_disable is shown as below:

Table 3-804. Function sdio_stop_readwait_disable

Function name	sdio_stop_readwait_disable
Function prototype	void sdio_stop_readwait_disable(void);
Function descriptions	disable the function that stop the read wait process(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the function that stop the read wait process(SD I/O only) */
sdio_stop_readwait_disable();
```

sdio_readwait_type_set

The description of sdio_readwait_type_set is shown as below:

Table 3-805. Function sdio_readwait_type_set

Function name	sdio_readwait_type_set
Function prototype	void sdio_readwait_type_set(uint32_t readwait_type);
Function descriptions	set the read wait type(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
readwait_type	SD I/O read wait type
SDIO_READWAITTYPE_CLK	read wait control by stopping SDIO_CLK

<i>SDIO_READWAITTYP E_DAT2</i>	read wait control using SDIO_DAT[2]
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the read wait type(SD I/O only) */
sdio_readwait_type_set(uint32_t readwait_type);
```

sdio_operation_enable

The description of sdio_operation_enable is shown as below:

Table 3-806. Function sdio_operation_enable

Function name	sdio_operation_enable
Function prototype	void sdio_operation_enable(void);
Function descriptions	enable the SD I/O mode specific operation(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the SD I/O mode specific operation(SD I/O only) */
sdio_operation_enable();
```

sdio_operation_disable

The description of sdio_operation_disable is shown as below:

Table 3-807. Function sdio_operation_disable

Function name	sdio_operation_disable
Function prototype	void sdio_operation_disable(void);
Function descriptions	disable the SD I/O mode specific operation(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the SD I/O mode specific operation(SD I/O only) */
void sdio_operation_disable();
```

sdio_suspend_enable

The description of sdio_suspend_enable is shown as below:

Table 3-808. Function sdio_suspend_enable

Function name	sdio_suspend_enable
Function prototype	void sdio_suspend_enable(void);
Function descriptions	enable the SD I/O suspend operation(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the SD I/O suspend operation(SD I/O only) */
sdio_suspend_enable();
```

sdio_suspend_disable

The description of sdio_suspend_disable is shown as below:

Table 3-809. Function sdio_suspend_disable

Function name	sdio_suspend_disable
Function prototype	void sdio_suspend_disable(void);
Function descriptions	disable the SD I/O suspend operation(SD I/O only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* disable the SD I/O suspend operation(SD I/O only) */
```

```
sdio_suspend_disable();
```

sdio_ceata_command_enable

The description of sdio_ceata_command_enable is shown as below:

Table 3-810. Function sdio_ceata_command_enable

Function name	sdio_ceata_command_enable
Function prototype	void sdio_ceata_command_enable(void);
Function descriptions	enable the CE-ATA command(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the CE-ATA command(CE-ATA only) */
```

```
sdio_ceata_command_enable();
```

sdio_ceata_command_disable

The description of sdio_ceata_command_disable is shown as below:

Table 3-811. Function sdio_ceata_command_disable

Function name	sdio_ceata_command_disable
Function prototype	void sdio_ceata_command_disable(void);
Function descriptions	disable the CE-ATA command(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CE-ATA command(CE-ATA only) */

sdio_ceata_command_disable();
```

sdio_ceata_interrupt_enable

The description of sdio_ceata_interrupt_enable is shown as below:

Table 3-812. Function sdio_ceata_interrupt_enable

Function name	sdio_ceata_interrupt_enable
Function prototype	void sdio_ceata_interrupt_enable(void);
Function descriptions	enable the CE-ATA interrupt(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the CE-ATA interrupt(CE-ATA only) */

sdio_ceata_interrupt_enable();
```

sdio_ceata_interrupt_disable

The description of sdio_ceata_interrupt_disable is shown as below:

Table 3-813. Function sdio_ceata_interrupt_disable

Function name	sdio_ceata_interrupt_disable
Function prototype	void sdio_ceata_interrupt_disable(void);
Function descriptions	disable the CE-ATA interrupt(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CE-ATA interrupt(CE-ATA only) */
```

```
sdio_ceata_interrupt_disable();
```

sdio_ceata_command_completion_enable

The description of sdio_ceata_command_completion_enable is shown as below:

Table 3-814. Function sdio_ceata_command_completion_enable

Function name	sdio_ceata_command_completion_enable
Function prototype	void sdio_ceata_command_completion_enable(void);
Function descriptions	enable the CE-ATA command completion signal(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the CE-ATA command completion signal(CE-ATA only) */
```

```
sdio_ceata_command_completion_enable();
```

sdio_ceata_command_completion_disable

The description of sdio_ceata_command_completion_disable is shown as below:

Table 3-815. Function sdio_ceata_command_completion_disable

Function name	sdio_ceata_command_completion_disable
Function prototype	void sdio_ceata_command_completion_disable(void);
Function descriptions	disable the CE-ATA command completion signal(CE-ATA only)
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CE-ATA command completion signal(CE-ATA only) */
```

```
sdio_ceata_command_completion_disable();
```

3.24. 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 [3.24.1](#), the SPI/I2S firmware functions are introduced in chapter [3.24.2](#).

3.24.1. Descriptions of Peripheral registers

SPI/I2S registers are listed in the table shown as below:

Table 3-816. 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

3.24.2. Descriptions of Peripheral functions

SPI/I2S firmware functions are listed in the table shown as below:

Table 3-817. 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_ckin_psc_config	configure I2S input clock 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

Function name	Function description
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
i2s_full_duplex_mode_config	configure i2s full duplex mode
spi_i2s_format_error_clear	clear TI mode format error flag status
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_crc_error_clear	clear SPI CRC error flag status
spi_ti_mode_enable	enable SPI TI mode
spi_ti_mode_disable	disable SPI TI 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_quad_io23_output_enable	enable quad wire SPI_IO2 and SPI_IO3 pin output
spi_quad_io23_output_disable	disable quad wire SPI_IO2 and SPI_IO3 pin output
spi_i2s_flag_get	get SPI and I2S flag status
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

Structure spi_parameter_struct

Table 3-818. Structure 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_FRAME_SIZE_16BIT, SPI_FRAME_SIZE_8BIT)
nss	SPI NSS control by hardware or software

Member name	Function description
	(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 3-819. 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
<i>SPIx</i>	x=0,1
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 3-820. 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 3-821. 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
Input parameter{in}	
spi_struct	SPI parameter initialization struct, the structure members can refer to members of the structure Table 3-818. Structure 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_FRAME_SIZE_8BIT;
spi_init_struct.clock_polarity_phase = SPI_CLOCK_POLARITY_HIGH_PHASE_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 3-822. 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
<i>SPIx</i>	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 */
```

```
spi_enable(SPI0);
```

spi_disable

The description of spi_disable is shown as below:

Table 3-823. 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
<i>SPIx</i>	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:


```
/* disable SPI0 */
```

```
spi_disable(SPI0);
```

i2s_init

The description of i2s_init is shown as below:

Table 3-824. Function i2s_init

Function name	i2s_init
Function prototype	void i2s_init(uint32_t spi_periph, uint32_t i2s_mode, uint32_t i2s_standard, uint32_t i2s_ckpl);
Function descriptions	initialize I2S peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
<i>SPIx</i>	x=1
Input parameter{in}	
i2s_mode	I2S operation mode
<i>I2S_MODE_SLAVETX</i>	I2S slave transmit mode
<i>I2S_MODE_SLAVEX</i>	I2S slave receive mode
<i>I2S_MODE_MASTERTX</i>	I2S master transmit mode
<i>I2S_MODE_MASTERR</i>	
<i>I2S_MODE_MASTERR</i>	I2S master receive mode
Input parameter{in}	
i2s_standard	I2S standard
<i>I2S_STD_PHILLIPS</i>	I2S phillips standard
<i>I2S_STD_MSB</i>	I2S MSB standard
<i>I2S_STD_LSB</i>	I2S LSB standard
<i>I2S_STD_PCMSHORT</i>	I2S PCM short standard
<i>I2S_STD_PCMLONG</i>	I2S PCM long standard
Input parameter{in}	
i2s_ckpl	I2S idle state clock polarity
<i>I2S_CKPL_LOW</i>	I2S clock polarity low level
<i>I2S_CKPL_HIGH</i>	I2S clock polarity high level
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize I2S1 */
```

```
i2s_init(SPI1, I2S_MODE_MASTERTX, I2S_STD_PHILLIPS, I2S_CKPL_LOW);
```

i2s_psc_config

The description of i2s_psc_config is shown as below:

Table 3-825. Function i2s_psc_config

Function name	i2s_psc_config
Function prototype	void i2s_psc_config(uint32_t spi_periph, uint32_t i2s_audiosample, uint32_t i2s_frameformat, uint32_t i2s_mckout);
Function descriptions	configure I2S prescaler
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
spi_periph	I2S peripheral
<i>SPIx</i>	x=1
Input parameter{in}	
i2s_audiosample	I2S audio sample rate
<i>I2S_AUDIOSAMPLE_8K</i>	audio sample rate is 8KHz
<i>I2S_AUDIOSAMPLE_11K</i>	audio sample rate is 11KHz
<i>I2S_AUDIOSAMPLE_16K</i>	audio sample rate is 16KHz
<i>I2S_AUDIOSAMPLE_22K</i>	audio sample rate is 22KHz
<i>I2S_AUDIOSAMPLE_32K</i>	audio sample rate is 32KHz
<i>I2S_AUDIOSAMPLE_44K</i>	audio sample rate is 44KHz
<i>I2S_AUDIOSAMPLE_48K</i>	audio sample rate is 48KHz
<i>I2S_AUDIOSAMPLE_96K</i>	audio sample rate is 96KHz
<i>I2S_AUDIOSAMPLE_192K</i>	audio sample rate is 192KHz
Input parameter{in}	
i2s_frameformat	I2S data length and channel length
<i>I2S_FRAMEFORMAT_DT16B_CH16B</i>	I2S data length is 16 bit and channel length is 16 bit
<i>I2S_FRAMEFORMAT_DT16B_CH32B</i>	I2S data length is 16 bit and channel length is 32 bit
<i>I2S_FRAMEFORMAT_DT24B_CH32B</i>	I2S data length is 24 bit and channel length is 32 bit

<i>I2S_FRAMEFORMAT_DT32B_CH32B</i>	I2S data length is 32 bit and channel length is 32 bit
Input parameter{in}	
i2s_mckout	I2S master clock output
<i>I2S_MCKOUT_ENABLER</i>	I2S master clock output enable
<i>I2S_MCKOUT_DISABLE</i>	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_ckin_psc_config

The description of i2s_ckin_psc_config is shown as below:

Table 3-826. Function i2s1_ckin_psc_config

Function name	i2s_ckin_psc_config
Function prototype	void i2s_ckin_psc_config(uint32_t input_clock, uint32_t i2s_audiosample, uint32_t i2s_frameformat, uint32_t i2s_mckout);
Function descriptions	configure I2S input clock prescaler
Precondition	-
The called functions	-
Input parameter{in}	
input_clock	I2S1 input clock
Input parameter{in}	
i2s_audiosample	I2S audio sample rate
<i>I2S_AUDIOSAMPLE_8K</i>	audio sample rate is 8KHz
<i>I2S_AUDIOSAMPLE_11K</i>	audio sample rate is 11KHz
<i>I2S_AUDIOSAMPLE_16K</i>	audio sample rate is 16KHz
<i>I2S_AUDIOSAMPLE_22K</i>	audio sample rate is 22KHz
<i>I2S_AUDIOSAMPLE_32K</i>	audio sample rate is 32KHz

<i>I2S_AUDIOSAMPLE_4</i> 4K	audio sample rate is 44KHz
<i>I2S_AUDIOSAMPLE_4</i> 8K	audio sample rate is 48KHz
<i>I2S_AUDIOSAMPLE_9</i> 6K	audio sample rate is 96KHz
<i>I2S_AUDIOSAMPLE_1</i> 92K	audio sample rate is 192KHz
Input parameter{in}	
i2s_frameformat	I2S data length and channel length
<i>I2S_FRAMEFORMAT_DT16B_CH16B</i>	I2S data length is 16 bit and channel length is 16 bit
<i>I2S_FRAMEFORMAT_DT16B_CH32B</i>	I2S data length is 16 bit and channel length is 32 bit
<i>I2S_FRAMEFORMAT_DT24B_CH32B</i>	I2S data length is 24 bit and channel length is 32 bit
<i>I2S_FRAMEFORMAT_DT32B_CH32B</i>	I2S data length is 32 bit and channel length is 32 bit
Input parameter{in}	
i2s_mckout	I2S master clock output
<i>I2S_MCKOUT_ENABL</i> E	I2S master clock output enable
<i>I2S_MCKOUT_DISABL</i> E	I2S master clock output disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure I2S1 input clock prescaler */
```

```
#define I2S1_INPUT_CK 50000000
```

```
i2s_psc_config(I2S1_INPUT_CK, I2S_AUDIOSAMPLE_8K, I2S_FRAMEFORMAT_DT16B_CH16B, I2S_MCKOUT_DISABLE);
```

i2s_enable

The description of i2s_enable is shown as below:

Table 3-827. 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
<i>SPI1</i>	I2S1 peripheral
<i>I2S1_ADD</i>	I2S1_ADD peripheral
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2S1*/
i2s_enable(SPI1);
```

i2s_disable

The description of i2s_disable is shown as below:

Table 3-828. 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
<i>SPI1</i>	I2S1 peripheral
<i>I2S1_ADD</i>	I2S1_ADD peripheral
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 3-829. 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
<i>SPIx</i>	x=0,1
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 3-830. 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
<i>SPIx</i>	x=0,1
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 3-831. 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
<i>SPIx</i>	x=0,1
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 3-832. 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
<i>SPIx</i>	x=0,1
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 3-833. 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
<i>SPIx</i>	x=0,1
Input parameter{in}	
dma	SPI DMA mode
<i>SPI_DMA_TRANSMIT</i>	SPI transmit data use DMA
<i>SPI_DMA_RECEIVE</i>	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 3-834. 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
<i>SPIx</i>	x=0,1
Input parameter{in}	
dma	SPI DMA mode
<i>SPI_DMA_TRANSMIT</i>	SPI transmit data use DMA
<i>SPI_DMA_RECEIVE</i>	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 3-835. 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
<i>SPIx</i>	x=0,1
Input parameter{in}	
frame_format	SPI frame size
<i>SPI_FRAME_SIZE_16BIT</i>	SPI frame size is 16 bits
<i>SPI_FRAME_SIZE_8BIT</i>	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_FRAME_SIZE_16BIT);
```

spi_i2s_data_transmit

The description of spi_i2s_data_transmit is shown as below:

Table 3-836. 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
<i>SPIx</i>	x=0,1

Input parameter{in}	
data	16-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 transmit data */

uint16_t spi0_send_array[] = {0x5050,0XA0A0};

spi_i2s_data_transmit(SPI0, spi0_send_array[0]);
```

spi_i2s_data_receive

The description of spi_i2s_data_receive is shown as below:

Table 3-837. 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
<i>SPIx</i>	x=0,1
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit data

Example:

```
/* SPI0 receive data */

uint16_t spi0_receive_data;

spi0_receive_data = spi_i2s_data_receive(SPI0);
```

spi_bidirectional_transfer_config

The description of spi_bidirectional_transfer_config is shown as below:

Table 3-838. 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
<i>SPIx</i>	x=0,1
Input parameter{in}	
transfer_direction	SPI transfer direction
<i>SPI_BIDIRECTIONAL_TRANSMIT</i>	SPI work in transmit-only mode
<i>SPI_BIDIRECTIONAL_RECEIVE</i>	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);
```

i2s_full_duplex_mode_config

The description of i2s_full_duplex_mode_config is shown as below:

Table 3-839. Function i2s_init

Function name	i2s_full_duplex_mode_config
Function prototype	void i2s_full_duplex_mode_config(uint32_t i2s_add_periph,uint32_t i2s_mode,uint32_t i2s_standard,uint32_t i2s_ckpl,uint32_t i2s_frameformat);
Function descriptions	configure i2s full duplex mode
Precondition	-
The called functions	-
Input parameter{in}	
i2s_add_periph	I2S_ADD peripheral
<i>I2Sx_ADD</i>	x=1
Input parameter{in}	
i2s_mode	I2S operation mode
<i>I2S_MODE_SLAVETX</i>	I2S slave transmit mode
<i>I2S_MODE_SLAVRX</i>	I2S slave receive mode
<i>I2S_MODE_MASTERTX</i>	I2S master transmit mode
<i>I2S_MODE_MASTERR</i>	I2S master receive mode

X	
Input parameter{in}	
i2s_standard	I2S standard
<i>I2S_STD_PHILLIPS</i>	I2S phillips standard
<i>I2S_STD_MSB</i>	I2S MSB standard
<i>I2S_STD_LSB</i>	I2S LSB standard
<i>I2S_STD_PCMSHORT</i>	I2S PCM short standard
<i>I2S_STD_PCMLONG</i>	I2S PCM long standard
Input parameter{in}	
i2s_ckpl	I2S idle state clock polarity
<i>I2S_CKPL_LOW</i>	I2S clock polarity low level
<i>I2S_CKPL_HIGH</i>	I2S clock polarity high level
Input parameter{in}	
i2s_frameformat	I2S data length and channel length
<i>I2S_FRAMEFORMAT_DT16B_CH16B</i>	I2S data length is 16 bit and channel length is 16 bit
<i>I2S_FRAMEFORMAT_DT16B_CH32B</i>	I2S data length is 16 bit and channel length is 32 bit
<i>I2S_FRAMEFORMAT_DT24B_CH32B</i>	I2S data length is 24 bit and channel length is 32 bit
<i>I2S_FRAMEFORMAT_DT32B_CH32B</i>	I2S data length is 32 bit and channel length is 32 bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure I2S1_ADD */
```

```
i2s_full_duplex_mode_config(I2S1_ADD,I2S_MODE_MASTERTX,I2S_STD_PHILLIPS,  
I2S_CKPL_HIGH, I2S_FRAMEFORMAT_DT16B_CH16B);
```

spi_i2s_format_error_clear

The description of spi_i2s_format_error_clear is shown as below:

Table 3-840. Function spi_i2s_format_error_clear

Function name	spi_i2s_format_error_clear
Function prototype	void spi_i2s_format_error_clear(uint32_t spi_periph, uint32_t flag);
Function descriptions	clear TI mode format error flag status
Precondition	-
The called functions	-
Input parameter{in}	

spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1
Input parameter{in}	
flag	SPI/I2S frame format error flag
<i>SPI_FLAG_FERR</i>	SPI TI mode frame format error
<i>I2S_FLAG_FERR</i>	I2S frame format error
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear SPI0 TI mode format error flag */
spi_i2s_format_error_clear(SPI0, SPI_FLAG_FERR);
```

spi_crc_polynomial_set

The description of spi_crc_polynomial_set is shown as below:

Table 3-841. 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
<i>SPIx</i>	x=0,1
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 3-842. Function `spi_crc_polynomial_get`

Function name	<code>spi_crc_polynomial_get</code>
Function prototype	<code>uint16_t spi_crc_polynomial_get(uint32_t spi_periph);</code>
Function descriptions	get SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
<code>spi_periph</code>	SPI peripheral
<code>SPIx</code>	x=0,1
Output parameter{out}	
-	-
Return value	
<code>uint16_t</code>	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 3-843. Function `spi_crc_on`

Function name	<code>spi_crc_on</code>
Function prototype	<code>void spi_crc_on(uint32_t spi_periph);</code>
Function descriptions	turn on SPI CRC function
Precondition	-
The called functions	-
Input parameter{in}	
<code>spi_periph</code>	SPI peripheral
<code>SPIx</code>	x=0,1
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 3-844. 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
<i>SPIx</i>	x=0,1
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 3-845. 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
<i>SPIx</i>	x=0,1
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 3-846. 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
<i>SPIx</i>	x=0,1
Input parameter{in}	
crc	SPI crc value
<i>SPI_CRC_TX</i>	get transmit crc value
<i>SPI_CRC_RX</i>	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_crc_error_clear

The description of spi_crc_error_clear is shown as below:

Table 3-847. 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
<i>SPIx</i>	x=0,1
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* clear SPI0 CRC error flag status */
```

```
spi_crc_error_clear(SPI0);
```

spi_ti_mode_enable

The description of spi_ti_mode_enable is shown as below:

Table 3-848. 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
<i>SPIx</i>	x=0,1
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 3-849. 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
<i>SPIx</i>	x=0,1
Output parameter{out}	
-	-
Return value	

Example:

```
/* disable SPI0 TI mode */  
  
spi_ti_mode_disable(SPI0);
```

spi_quad_enable

The description of spi_quad_enable is shown as below:

Table 3-850. 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
<i>SPIx</i>	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 3-851. 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
<i>SPIx</i>	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 3-852. 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
<i>SPIx</i>	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 3-853. 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
<i>SPIx</i>	x=0
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable SPI0 quad wire read */
```

```
spi_quad_read_enable(SPI0);
```

spi_quad_io23_output_enable

The description of spi_quad_io23_output_enable is shown as below:

Table 3-854. Function spi_quad_io23_output_enable

Function name	spi_quad_io23_output_enable
Function prototype	void spi_quad_io23_output_enable(uint32_t spi_periph);
Function descriptions	enable SPI_IO2 and SPI_IO3 pin output
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 SPI_IO2 and SPI_IO3 pin output */
```

```
spi_quad_io23_output_enable(SPI0);
```

spi_quad_io23_output_disable

The description of spi_quad_io23_output_disable is shown as below:

Table 3-855. Function spi_quad_io23_output_disable

Function name	spi_quad_io23_output_disable
Function prototype	void spi_quad_io23_output_disable(uint32_t spi_periph);
Function descriptions	disable SPI_IO2 and SPI_IO3 pin output
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0
Output parameter{out}	
-	-
Return value	

Example:

```
/* disable SPI0 SPI_IO2 and SPI_IO3 pin output */
```

```
spi_quad_io23_output_disable(SPI0);
```

spi_i2s_flag_get

The description of spi_i2s_flag_get is shown as below:

Table 3-856. 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
<i>SPIx</i>	x=0,1
Input parameter{in}	
flag	SPI/I2S flag status
<i>SPI_FLAG_TBE</i>	transmit buffer empty flag
<i>SPI_FLAG_RBNE</i>	receive buffer not empty flag
<i>SPI_FLAG_TRANS</i>	transmit on-going flag
<i>SPI_I2S_INT_FLAG_RXORERR</i>	receive overrun error flag
<i>SPI_FLAG_CONFERR</i>	mode config error flag
<i>SPI_FLAG_CRCERR</i>	CRC error flag
<i>SPI_FLAG_FREE</i>	format error flag
<i>I2S_FLAG_TBE</i>	transmit buffer empty flag
<i>I2S_FLAG_RBNE</i>	receive buffer not empty flag
<i>I2S_FLAG_TRANS</i>	transmit on-going flag
<i>I2S_FLAG_RXORERR</i>	overrun error flag
<i>I2S_FLAG_TXURERR</i>	underrun error flag
<i>I2S_FLAG_CH</i>	channel side flag
<i>I2S_FLAG_FERR</i>	format error 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_i2s_interrupt_enable

The description of spi_i2s_interrupt_enable is shown as below:

Table 3-857. 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
<i>SPIx</i>	x=0,1
Input parameter{in}	
interrupt	SPI/I2S interrupt
<i>SPI_I2S_INT_TBE</i>	transmit buffer empty interrupt
<i>SPI_I2S_INT_RBNE</i>	receive buffer not empty interrupt
<i>SPI_I2S_INT_ERR</i>	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 3-858. 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
<i>SPIx</i>	x=0,1

Input parameter{in}	
interrupt	SPI/I2S interrupt
<i>SPI_I2S_INT_TBE</i>	transmit buffer empty interrupt
<i>SPI_I2S_INT_RBNE</i>	receive buffer not empty interrupt
<i>SPI_I2S_INT_ERR</i>	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 3-859. 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
<i>SPIx</i>	x=0,1
Input parameter{in}	
interrupt	SPI/I2S interrupt flag status
<i>SPI_I2S_INT_FLAG_TBE</i>	transmit buffer empty interrupt
<i>SPI_I2S_INT_FLAG_RBNE</i>	receive buffer not empty interrupt
<i>SPI_I2S_INT_FLAG_RXORERR</i>	overrun interrupt
<i>SPI_INT_FLAG_CONFERR</i>	config error interrupt
<i>SPI_INT_FLAG_CRCERR</i>	CRC error interrupt
<i>I2S_INT_FLAG_TXURERR</i>	underrun error interrupt
<i>SPI_I2S_INT_FLAG_F</i>	format error interrupt

<i>ERR</i>	
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++]);
}

```

3.25. SQPI

Serial/Quad Parallel Interface (SQPI) is a controller for external serial/dual/quad parallel interface memory peripheral. The SQPI registers are listed in chapter [3.25.1](#), the SQPI firmware functions are introduced in chapter [3.25.2](#).

3.25.1. Descriptions of Peripheral registers

SQPI registers are listed in the table shown as below:

Table 3-860. SQPI Registers

Registers	Descriptions
SQPI_INIT	SQPI initial register
SQPI_RCMD	SQPI read command register
SQPI_WCMD	SQPI write command register
SQPI_IDL	SQPI ID low register
SQPI_IDH	SQPI ID high register

3.25.2. Descriptions of Peripheral functions

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

Table 3-861. SQPI firmware function

Function name	Function description
sqpi_deinit	reset SQPI peripheral
sqpi_struct_para_init	initialize the parameters of SQPI struct with the default values
sqpi_init	initialize SQPI peripheral parameter
sqpi_read_id_command	send SQPI read ID command

Function name	Function description
sqpi_special_command	send SQPI special command
sqpi_read_command_config	configure SQPI read command
sqpi_write_command_config	configure SQPI write command
sqpi_low_id_receive	SQPI receive low ID
sqpi_high_id_receive	SQPI receive high ID

Structure sqpi_parameter_struct

Table 3-862. sqpi_parameter_struct

Member name	Function description
polarity	SQPI sample polarity (SQPI_SAMPLE_POLARITY_RISING, SQPI_SAMPLE_POLARITY_FALLING)
id_length	external memory ID length (QSPI_ID_LENGTH_n_BITS (n=8,16,32,64))
addr_bit	bit number of SPI PSRAM address phase (0x00 - 0x1F)
clk_div	clock divider for SQPI output clock (0x01 - 0x3F)
cmd_bit	bit number of SQPI controller command phase (QSPI_CMDBIT_n_BITS (n=4,8,16))

sqpi_deinit

The description of sqpi_deinit is shown as below:

Table 3-863. Function sqpi_deinit

Function name	sqpi_deinit
Function prototype	void sqpi_deinit(void);
Function descriptions	reset SQPI peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset SQPI */
sqpi_deinit();
```

sqpi_struct_para_init

The description of sqpi_struct_para_init is shown as below:

Table 3-864. Function sqpi_struct_para_init

Function name	sqpi_struct_para_init
Function prototype	void sqpi_struct_para_init(sqpi_parameter_struct* sqpi_struct);
Function descriptions	initialize the parameters of SQPI struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
*sqpi_struct	a sqpi_parameter_struct address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the parameters of SPI */
sqpi_parameter_struct sqpi_init_struct;
sqpi_struct_para_init(&sqpi_init_struct);
```

sqpi_init

The description of sqpi_init is shown as below:

Table 3-865. Function sqpi_init

Function name	sqpi_init
Function prototype	void sqpi_init(sqpi_parameter_struct* sqpi_struct);
Function descriptions	initialize SQPI peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
sqpi_struct	SPI parameter initialization struct, the structure members can refer to members of the structure Table 3-862. sqpi_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize SQPI */
sqpi_parameter_struct sqpi_init_struct;
```

```
smpi_struct->polarity = SQPI_SAMPLE_POLARITY_RISING;
```

```
smpi_struct->id_length = QSPI_ID_LENGTH_32_BITS;
```

```
smpi_struct->addr_bit = 24U;
```

```
smpi_struct->clk_div = 2U;
```

```
smpi_struct->cmd_bit = QSPI_CMDBIT_8_BITS;
```

```
smpi_init(&smpi_init_struct);
```

smpi_read_id_command

The description of smpi_read_id_command is shown as below:

Table 3-866. Function smpi_read_id_command

Function name	smpi_read_id_command
Function prototype	void smpi_read_id_command (void);
Function descriptions	send SQPI read ID command
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* send SQPI read ID command */
```

```
smpi_read_id_command ();
```

smpi_special_command

The description of smpi_special_command is shown as below:

Table 3-867. Function smpi_special_command

Function name	smpi_special_command
Function prototype	void smpi_special_command(void);
Function descriptions	send SQPI special command
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* send SQPI special command */
sqpi_special_command ();
```

sqpi_read_command_config

The description of sqpi_read_command_config is shown as below:

Table 3-868. Function sqpi_read_command_config

Function name	sqpi_read_command_config
Function prototype	void sqpi_read_command_config(uint32_t rmode, uint32_t rwaitcycle, uint32_t rcmd);
Function descriptions	configure SQPI read command
Precondition	-
The called functions	-
Input parameter{in}	
rmode	SQPI read command mode
QSPI_MODE_SSQ	SSQ mode
QSPI_MODE_SSS	SSS mode
QSPI_MODE_SQQ	SQQ mode
QSPI_MODE_QQQ	QQQ mode
QSPI_MODE_SSD	SSD mode
QSPI_MODE_SDD	SDD mode
Input parameter{in}	
rwaitcycle	SQPI read wait cycle (0x00 – 0x1F)
Input parameter{in}	
rcmd	SQPI read command(0x00 – 0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SQPI read command */
sqpi_read_command_config(QSPI_MODE_SSS,0x00,0x9f);
```

sqpi_write_command_config

The description of sqpi_write_command_config is shown as below:

Table 3-869. Function `sqpi_write_command_config`

Function name	<code>sqpi_write_command_config</code>
Function prototype	<code>void sqpi_write_command_config(uint32_t wmode, uint32_t wwaitcycle, uint32_t wcmd);</code>
Function descriptions	configure SQPI write command
Precondition	-
The called functions	-
Input parameter{in}	
wmode	SQPI write command mode
<code>QSPI_MODE_SSQ</code>	SSQ mode
<code>QSPI_MODE_SSS</code>	SSS mode
<code>QSPI_MODE_SQQ</code>	SQQ mode
<code>QSPI_MODE_QQQ</code>	QQQ mode
<code>QSPI_MODE_SSD</code>	SSD mode
<code>QSPI_MODE_SDD</code>	SDD mode
Input parameter{in}	
wwaitcycle	SQPI write wait cycle (0x00 – 0x1F)
Input parameter{in}	
wcmd	SQPI write command(0x00 – 0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SQPI write command */
sqpi_write_command_config(QSPI_MODE_SSS,0x00,0x9f);
```

`sqpi_low_id_receive`

The description of `sqpi_low_id_receive` is shown as below:

Table 3-870. Function `sqpi_low_id_receive`

Function name	<code>sqpi_low_id_receive</code>
Function prototype	<code>uint32_t sqpi_low_id_receive(void);</code>
Function descriptions	get low ID value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

uint32_t	32-bit IDvalue (0-0xFFFF)
----------	---------------------------

Example:

```
/* get SQPI low ID value */

uint32_t val;

val = sqpi_low_id_receive ();
```

sqpi_high_id_receive

The description of sqpi_high_id_receive is shown as below:

Table 3-871. Function sqpi_low_id_receive

Function name	sqpi_high_id_receive
Function prototype	uint32_t sqpi_high_id_receive(void);
Function descriptions	get high ID value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit ID value (0-0xFFFF)

Example:

```
/* get SQPI high ID value */

uint32_t val;

val = sqpi_high_id_receive ();
```

3.26. SYSCFG

The SYSCFG registers are listed in chapter [3.26.1](#), the SYSCFG firmware functions are introduced in chapter [3.26.2](#).

3.26.1. Descriptions of Peripheral registers

SYSCFG registers are listed in the table shown as below:

Table 3-872. SYSCFG registers

Registers	Descriptions
SYSCFG_CFG0	SYSCFG configuration register 0
SYSCFG_EXTISS0	EXTI sources selection register 0

Registers	Descriptions
SYSCFG_EXTISS1	EXTI sources selection register 1
SYSCFG_EXTISS2	EXTI sources selection register 2
SYSCFG_EXTISS3	EXTI sources selection register 3
SYSCFG_CPCTL	I/O compensation control register
SYSCFG_SECCFG	SYSCFG secure configuration register
SYSCFG_FPUINTMSK	FPU interrupt mask register
SYSCFG_CNSLOCK	SYSCFG CPU non-secure lock register
SYSCFG_CSLOCK	SYSCFG CPU secure lock register
SYSCFG_CFG1	SYSCFG configuration register 1
SYSCFG_SCS	SYSCFG SRAM1 control and status register
SYSCFG_SKEY	SYSCFG SRAM1 key register
SYSCFG_SWP0	SYSCFG SRAM1 write protection register 0
SYSCFG_SWP1	SYSCFG SRAM1 write protection register 1
SYSCFG_GSSACMD	SYSCFG GSSA command register

3.26.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

Table 3-873.SYSCFG firmware function

Function name	Function description
syscfg_deinit	reset the SYSCFG registers
syscfg_exti_line_config	configure the GPIO pin as EXTI Line
compensation_pwn_mode_enable	enable the compensation cell
compensation_pwn_mode_disable	disable the compensation cell
syscfg_clock_access_security_config	configure SYSCFG clock control security
classb_access_security_config	configure ClassB security
sram1_access_security_config	configure SRAM1 security
fpu_access_security_config	configure FPU security
vtor_ns_write_disable	disable VTOR_NS register write
mpu_ns_write_disable	disable Non-secure MPU registers write
vtors_aircr_write_disable	disable VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write
mpu_s_write_disable	disable secure MPU registers writes
sau_write_disable	disable SAU registers write
syscfg_lock_config	connect TIMER0/15/16 break input to the selected parameter
gssacmd_write_data	write data to GSSA
sram1_erase	enable sram1 erase
sram1_unlock	unlock the write protection of the SRAM1ERS bit in the

Function name	Function description
	SYSCFG_SCS register
sram1_lock	lock the write protection of the SRAM1ERS bit in the SYSCFG_SCS register
sram1_write_protect_0_31	SRAM1 write protect range from page0 to page31
sram1_write_protect_32_63	SRAM1 write protect range from page32 to page63
compensation_ready_flag_get	get the compensation ready flag
sram1_bsy_flag_get	get SRAM1 erase busy flag
fpv_interrupt_enable	enable floating point unit interrupt
fpv_interrupt_disable	disable floating point unit interrupt

syscfg_deinit

The description of syscfg_deinit is shown as below:

Table 3-874. Function syscfg_deinit

Function name	syscfg_deinit
Function prototype	void syscfg_deinit(void);
Function descriptions	reset the SYSCFG registers
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset SYSCFG */
syscfg_deinit();
```

syscfg_exti_line_config

The description of syscfg_exti_line_config is shown as below:

Table 3-875. Function syscfg_exti_line_config

Function name	syscfg_exti_line_config
Function prototype	void syscfg_exti_line_config(uint8_t exti_port, uint8_t exti_pin);
Function descriptions	configure the GPIO pin as EXTI Line
Precondition	-
The called functions	-
Input parameter{in}	
exti_port	specify the GPIO port used in EXTI

EXTI_SOURCE_GPIOx	EXTI GPIO port (x = A, B, C)
Input parameter{in}	
exti_pin	specify the EXTI line
EXTI_SOURCE_PINx	EXTI GPIO pin (GPIOA x = 0..15, GPIOB x = 0..15, GPIOC x = 0..8)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the GPIO pin as EXTI Line */
syscfg_exti_line_config(EXTI_SOURCE_GPIOA, EXTI_SOURCE_PIN1);
```

compensation_pwdn_mode_enable

The description of compensation_pwdn_mode_enable is shown as below:

Table 3-876. Function compensation_pwdn_mode_enable

Function name	compensation_pwdn_mode_enable
Function prototype	void compensation_pwdn_mode_enable(void);
Function descriptions	enable the compensation cell
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the compensation cell */
compensation_pwdn_mode_enable();
```

compensation_pwdn_mode_disable

The description of compensation_pwdn_mode_disable is shown as below:

Table 3-877. Function compensation_pwdn_mode_disable

Function name	compensation_pwdn_mode_disable
Function prototype	void compensation_pwdn_mode_disable(void)
Function descriptions	disable the compensation cell
Precondition	-
The called functions	-

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

Example:

```
/* disable the compensation cell */
compensation_pwdn_mode_disable();
```

syscfg_clock_access_security_config

The description of syscfg_clock_access_security_config is shown as below:

Table 3-878. Function syscfg_clock_access_security_config

Function name	syscfg_clock_access_security_config
Function prototype	void syscfg_clock_access_security_config(uint32_t access_mode);
Function descriptions	configure SYSCFG clock control security
Precondition	-
The called functions	-
Input parameter{in}	
access_mode	secure access or secure and non-secure access
SYSCFG_SECURE_ACCESS	SYSCFG configuration clock in RCU registers can be written by secure access only
SYSCFG_SECURE_NONSECURE_ACCESS	SYSCFG configuration clock in RCU registers can be written by secure and nonsecure access
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SYSCFG clock control security */
syscfg_clock_access_security_config(SYSCFG_SECURE_ACCESS);
```

classb_access_security_config

The description of classb_access_security_config is shown as below:

Table 3-879. Function classb_access_security_config

Function name	classb_access_security_config
Function prototype	void classb_access_security_config(uint32_t access_mode);
Function descriptions	configure ClassB security

Precondition	-
The called functions	-
Input parameter{in}	
access_mode	secure access or secure and non-secure access
<i>CLASSB_SECURE_ACCESS</i>	SYSCFG_CFG1 register can be written by secure access only
<i>CLASSB_SECURE_NONSECURE_ACCESS</i>	SYSCFG_CFG1 register can be written by secure and non-secure access
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ClassB security */
classb_access_security_config(CLASSB_SECURE_ACCESS);
```

sram1_access_security_config

The description of sram1_access_security_config is shown as below:

Table 3-880. Function sram1_access_security_config

Function name	sram1_access_security_config
Function prototype	void sram1_access_security_config(uint32_t access_mode)
Function descriptions	configure SRAM1 security
Precondition	-
The called functions	-
Input parameter{in}	
access_mode	secure access or secure and non-secure access
<i>SRAM1_SECURE_ACCESS</i>	SYSCFG_SKEY, SYSCFG_SCS and SYSCFG_SWPx registers can be written by secure only
<i>SRAM1_SECURE_NONSECURE_ACCESS</i>	SYSCFG_SKEY, SYSCFG_SCS and SYSCFG_SWPx registers can be written by secure and non-secure access
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SRAM1 security */
sram1_access_security_config(SRAM1_SECURE_ACCESS);
```

fpu_access_security_config

The description of fpu_access_security_config is shown as below:

Table 3-881. Function fpu_access_security_config

Function name	fpu_access_security_config
Function prototype	void fpu_access_security_config(uint32_t access_mode);
Function descriptions	configure FPU security
Precondition	-
The called functions	-
Input parameter{in}	
access_mode	secure access or secure and non-secure access
<i>FPU_SECURE_ACCE</i> <i>SS</i>	SYSCFG_FPUINTMSK register can be written by secure access only
<i>FPU_SECURE_NONS</i> <i>ECURE_ACCESS</i>	SYSCFG_FPUINTMSK register can be written by secure and non-secure access
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure FPU security */
fpu_access_security_config(FPU_SECURE_ACCESS);
```

vtor_ns_write_disable

The description of vtor_ns_write_disable is shown as below:

Table 3-882. Function vtor_ns_write_disable

Function name	vtor_ns_write_disable
Function prototype	void vtor_ns_write_disable(void);
Function descriptions	disable VTOR_NS register write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable VTOR_NS register write */
```

vtor_ns_write_disable();

mpu_ns_write_disable

The description of mpu_ns_write_disable is shown as below:

Table 3-883. Function mpu_ns_write_disable

Function name	mpu_ns_write_disable
Function prototype	void mpu_ns_write_disable(void)
Function descriptions	disable Non-secure MPU registers write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable Non-secure MPU registers write */
```

```
mpu_ns_write_disable();
```

vtors_aircr_write_disable

The description of vtors_aircr_write_disable is shown as below:

Table 3-884. Function vtors_aircr_write_disable

Function name	vtors_aircr_write_disable
Function prototype	void vtors_aircr_write_disable(void)
Function descriptions	disable VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write */
```

```
vtors_aircr_write_disable();
```

vtors_aircr_write_disable

The description of vtors_aircr_write_disable is shown as below:

Table 3-885. Function vtors_aircr_write_disable

Function name	vtors_aircr_write_disable
Function prototype	void mpu_s_write_disable(void);
Function descriptions	disable secure MPU registers writes
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable secure MPU registers writes */
mpu_s_write_disable();
```

sau_write_disable

The description of sau_write_disable is shown as below:

Table 3-886. Function sau_write_disable

Function name	sau_write_disable
Function prototype	void sau_write_disable(void);
Function descriptions	disable SAU registers write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SAU registers write */
sau_write_disable();
```

syscfg_lock_config

The description of syscfg_lock_config is shown as below:

Table 3-887. Function syscfg_lock_config

Function name	syscfg_lock_config
Function prototype	void syscfg_lock_config(uint32_t syscfg_lock);
Function descriptions	connect TIMER0/15/16 break input to the selected parameter
Precondition	-
The called functions	-
Input parameter{in}	
syscfg_lock	specify the parameter to be connected
SYSCFG_LOCK_LOCKUP	Cortex-M3 lockup output connected to the break input
SYSCFG_LOCK_LVD	LVD interrupt connected to the break input
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* connect TIMER0/15/16 break input to the selected parameter */
syscfg_lock_config(SYSCFG_LOCK_LOCKUP);
```

gssacmd_write_data

The description of gssacmd_write_data is shown as below:

Table 3-888. Function gssacmd_write_data

Function name	gssacmd_write_data
Function prototype	void gssacmd_write_data(uint32_t data);
Function descriptions	write data to GSSA
Precondition	-
The called functions	-
Input parameter{in}	
data	the data to be written in to GSSA(0x00-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write data to GSSA */
gssacmd_write_data(0x0C);
```

sram1_erase

The description of sram1_erase is shown as below:

Table 3-889. Function sram1_erase

Function name	sram1_erase
Function prototype	ErrStatus sram1_erase(void);
Function descriptions	enable sram1 erase
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
/* enable sram1 erase */
```

```
ErrStatus state;
```

```
state = sram1_erase();
```

sram1_unlock

The description of sram1_unlock is shown as below:

Table 3-890. Function sram1_unlock

Function name	sram1_unlock
Function prototype	void sram1_unlock(void);
Function descriptions	unlock the write protection of the SRAM1ERS bit in the SYSCFG_SCS register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the write protection of the SRAM1ERS bit in the SYSCFG_SCS register */
```

```
sram1_unlock();
```


sram1_lock

The description of sram1_lock is shown as below:

Table 3-891. Function sram1_lock

Function name	sram1_lock
Function prototype	void sram1_lock(void);
Function descriptions	lock the write protection of the SRAM1ERS bit in the SYSCFG_SCS register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the write protection of the SRAM1ERS bit in the SYSCFG_SCS register */
sram1_lock();
```

sram1_write_protect_0_31

The description of sram1_write_protect_0_31 is shown as below:

Table 3-892. Function sram1_write_protect_0_31

Function name	sram1_write_protect_0_31
Function prototype	void sram1_write_protect_0_31(uint32_t page);
Function descriptions	SRAM1 write protect range from page0 to page31
Precondition	-
The called functions	-
Input parameter{in}	
page	specify the page of SRAM1 to protect
<i>SRAM1_PAGEx_WRITE_PROTECT</i>	enable write protection of SRAM1 page x (x=0,1,...31)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SRAM1 write protect page1 */
sram1_write_protect_0_31(SRAM1_PAGE1_WRITE_PROTECT);
```

sram1_write_protect_32_63

The description of sram1_write_protect_32_63 is shown as below:

Table 3-893. Function sram1_write_protect_32_63

Function name	sram1_write_protect_32_63
Function prototype	void sram1_write_protect_32_63(uint32_t page);
Function descriptions	SRAM1 write protect range from page32 to page63
Precondition	-
The called functions	-
Input parameter{in}	
page	specify the page of SRAM1 to protect
<i>SRAM1_PAGEx_WRITE_PROTECT</i>	enable write protection of SRAM1 page x (x=32,33,...63)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SRAM1 write protect page32 */
sram1_write_protect_32_63(SRAM1_PAGE32_WRITE_PROTECT);
```

compensation_ready_flag_get

The description of compensation_ready_flag_get is shown as below:

Table 3-894. Function compensation_ready_flag_get

Function name	compensation_ready_flag_get
Function prototype	FlagStatus compensation_ready_flag_get(void);
Function descriptions	get the compensation ready flag
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	RESET or SET

Example:

```
/* get the compensation ready flag */
compensation_ready_flag_get();
```

sram1_bsy_flag_get

The description of sram1_bsy_flag_get is shown as below:

Table 3-895. Function sram1_bsy_flag_get

Function name	sram1_bsy_flag_get
Function prototype	FlagStatus sram1_bsy_flag_get(void);
Function descriptions	get SRAM1 erase busy flag
Precondition	-
The called functions	-
Input parameter{in}	
Output parameter{out}	
-	-
Return value	
FlagStatus	SET of RESET

Example:

```
/* get SRAM1 erase busy flag */
```

```
FlagStatus state;
```

```
State = sram1_bsy_flag_get();
```

fpu_interrupt_enable

The description of fpu_interrupt_enable is shown as below:

Table 3-896. Function fpu_interrupt_enable

Function name	fpu_interrupt_enable
Function prototype	void fpu_interrupt_enable(uint32_t interrupt);
Function descriptions	enable floating point unit interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	interrupt type
<i>INVALID_OPERATION_INT</i>	invalid operation Interrupt
<i>DEVIDE_BY_ZERO_INT</i>	divide-by-zero interrupt
<i>UNDERFLOW_INT</i>	underflow interrupt
<i>OVERFLOW_INT</i>	overflow interrupt
<i>INPUT_ABNORMAL_INT</i>	input abnormal interrupt
<i>INEXACT_INT</i>	inexact interrupt (interrupt disable at reset)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable floating point unit interrupt */
```

```
fpu_interrupt_enable(INVALID_OPERATION_INT);
```

fpu_interrupt_disable

The description of fpu_interrupt_disable is shown as below:

Table 3-897. Function fpu_interrupt_disable

Function name	fpu_interrupt_disable
Function prototype	void fpu_interrupt_disable(uint32_t interrupt);
Function descriptions	disable floating point unit interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	interrupt type
INVALID_OPERATION_INT	invalid operation Interrupt
DEVIDE_BY_ZERO_INTERRUPT	divide-by-zero interrupt
UNDERFLOW_INT	underflow interrupt
OVERFLOW_INT	overflow interrupt
INPUT_ABNORMAL_INTERRUPT	input abnormal interrupt
INEXACT_INT	inexact interrupt (interrupt disable at reset)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable floating point unit interrupt */
```

```
fpu_interrupt_disable(INVALID_OPERATION_INT);
```

3.27. 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), general level0 timer (TIMERx, x=1, 2, 3, 4), general level4 timer (TIMERx, x=15, 16), basic timer (TIMERx, x=5). The specific functions of different types of timer are different. The TIMER registers are listed in chapter [3.27.1](#), the TIMER firmware functions are introduced in chapter [3.27.2](#).

3.27.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

Table 3-898. 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
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

3.27.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

Table 3-899. 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

Function name	Function description
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
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

Function name	Function description
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 value
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
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_flag_get	get TIMER flags
timer_flag_clear	clear TIMER flags

Function name	Function description
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

Structure timer_parameter_struct

Table 3-900. 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
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 3-901. 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, 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 3-902. 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)

Member name	Function description
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 3-903. 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 3-904. 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
<i>TIMERx(x=0..5, 15, 16)</i>	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 3-905. 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 3-900. 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 3-906. 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
<i>TIMERx(x=0..5, 15, 16)</i>	TIMER peripheral selection
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to Table 3-900. 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 3-907. 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
<i>TIMERx(x=0..5,15,16)</i>	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 3-908. 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..5, 15, 16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMERO */
```

```
timer_disable(TIMERO);
```

timer_auto_reload_shadow_enable

The description of timer_auto_reload_shadow_enable is shown as below:

Table 3-909. 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..5, 15, 16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMERO auto reload shadow function */
```

```
timer_auto_reload_shadow_enable(TIMERO);
```

timer_auto_reload_shadow_disable

The description of timer_auto_reload_shadow_disable is shown as below:

Table 3-910. 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
<i>TIMERx(x=0..5, 15, 16)</i>	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 3-911. 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
<i>TIMERx(x=0..5, 15, 16)</i>	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 3-912. 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
<i>TIMERx(x=0..5, 15, 16)</i>	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 3-913. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
aligned	alignment mode
<i>TIMER_COUNTER_EDGE</i>	No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit.
<i>TIMER_COUNTER_COUNTER_DOWN</i>	Center-aligned and counting down assert mode. The counter counts under center aligned and channel is configured in output mode (CHxMS=00 in <i>TIMERx_CHCTL0</i> register). Only when the counter is counting down, compare interrupt flag of channels can be set.
<i>TIMER_COUNTER_COUNTER_UP</i>	Center-aligned and counting up assert mode. The counter counts under center aligned and channel is configured in output mode (CHxMS=00 in <i>TIMERx_CHCTL0</i> register). Only when the counter is counting up, compare interrupt flag of channels can be set.

<i>TIMER_COUNTER_CENTER_BOTH</i>	Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMEx_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 3-914. 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
<i>TIMEx(x=0..4)</i>	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 3-915. 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
<i>TIMERx(x=0..4)</i>	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 3-916. Function timer_prescaler_config

Function name	timer_prescaler_config
Function prototype	void timer_prescaler_config(uint32_t timer_periph, uint16_t prescaler, uint32_t pscreload);
Function descriptions	configure TIMER prescaler
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0..5,15,16)</i>	TIMER peripheral selection
Input parameter{in}	
prescaler	prescaler value (0~65535)
Input parameter{in}	
pscreload	prescaler reload mode
<i>TIMER_PSC_RELOAD_NOW</i>	the prescaler is loaded right now
<i>TIMER_PSC_RELOAD_UPDATE</i>	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 3-917. 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
<i>TIMERx(x=0, 15, 16)</i>	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 3-918. Function timer_autoreload_value_config

Function name	timer_autoreload_value_config
Function prototype	void timer_autoreload_value_config(uint32_t timer_periph, uint32_t autoreload);
Function descriptions	configure TIMER autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0..5, 15, 16)</i>	TIMER peripheral selection
Input parameter{in}	
autoreload	the counter auto-reload value (0-0xFFFFFFFF)
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 3-919. Function timer_counter_value_config

Function name	timer_counter_value_config
Function prototype	void timer_counter_value_config(uint32_t timer_periph, uint32_t counter);
Function descriptions	configure TIMER counter register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0..5, 15, 16)</i>	TIMER peripheral selection
Input parameter{in}	
counter	the counter value (0-0xFFFFFFFF)
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 3-920. 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
<i>TIMERx(x=0..5, 15, 16)</i>	TIMER peripheral selection
Output parameter{out}	

-	-
Return value	
uint32_t	counter value(0~0xFFFFFFFF)

Example:

```
/* read TIMERO counter value */

uint32_t i = 0;

i = timer_counter_read(TIMERO);
```

timer_prescaler_read

The description of timer_prescaler_read is shown as below:

Table 3-921. 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
TIMEx(x=0..5, 15, 16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	prescaler register value (0x0000~0xFFFF)

Example:

```
/* read TIMERO prescaler value */

uint16_t i = 0;

i = timer_prescaler_read(TIMERO);
```

timer_single_pulse_mode_config

The description of timer_single_pulse_mode_config is shown as below:

Table 3-922. 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
<i>TIMERx(x=0..5, 15, 16)</i>	TIMER peripheral selection
Input parameter{in}	
spmode	pulse mode
<i>TIMER_SP_MODE_SINGLE</i>	single pulse mode
<i>TIMER_SP_MODE_REPETITIVE</i>	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 3-923. 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
<i>TIMERx(x=0..5, 15, 16)</i>	TIMER peripheral selection
Input parameter{in}	
update	update source
<i>TIMER_UPDATE_SRC_GLOBAL</i>	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
<i>TIMER_UPDATE_SRC_REGULAR</i>	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 3-924. 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..5,15,16)
TIMER_DMA_CH0D	channel 0 DMA enable, TIMERx(x=0..4,,15,16)
TIMER_DMA_CH1D	channel 1 DMA enable, TIMERx(x=0..4)
TIMER_DMA_CH2D	channel 2 DMA enable, TIMERx(x=0..4)
TIMER_DMA_CH3D	channel 3 DMA enable, TIMERx(x=0..4)
TIMER_DMA_CMTD	commutation DMA request enable, TIMERx(x=0,15,16)
TIMER_DMA_TRGD	trigger DMA enable, TIMERx(x=0..4)
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 3-925. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
dma	timer DMA source disable
<i>TIMER_DMA_UPD</i>	update DMA disable, TIMERx(x=0..5,15,16)
<i>TIMER_DMA_CH0D</i>	channel 0 DMA disable, TIMERx(x=0..4,15,16)
<i>TIMER_DMA_CH1D</i>	channel 1 DMA disable, TIMERx(x=0..4)
<i>TIMER_DMA_CH2D</i>	channel 2 DMA disable, TIMERx(x=0..4)
<i>TIMER_DMA_CH3D</i>	channel 3 DMA disable, TIMERx(x=0..4)
<i>TIMER_DMA_CMTD</i>	commutation DMA request disable, TIMERx(x=0,15,16)
<i>TIMER_DMA_TRGD</i>	trigger DMA disable, TIMERx(x=0..4)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMERO0 update DMA */
```

```
timer_dma_disable(TIMERO0, TIMER_DMA_UPD);
```

timer_channel_dma_request_source_select

The description of timer_channel_dma_request_source_select is shown as below:

Table 3-926. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
dma_request	channel DMA request source selection
<i>TIMER_DMAREQUEST_CHANNELEVENT</i>	DMA request of channel n is sent when channel y event occurs

<i>TIMER_DMAREQUEST_UPDATEEVENT</i>	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 3-927. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
dma_baseaddr	DMA transfer access start address
<i>TIMER_DMACFG_DMA_TA_CTL0</i>	DMA transfer address is TIMER_CTL0, TIMERx(x=0..4)
<i>TIMER_DMACFG_DMA_TA_CTL1</i>	DMA transfer address is TIMER_CTL1, TIMERx(x=0..4)
<i>TIMER_DMACFG_DMA_TA_SMCFG</i>	DMA transfer address is TIMER_SMCFG, TIMERx(x=0..4)
<i>TIMER_DMACFG_DMA_TA_DMAINTEN</i>	DMA transfer address is TIMER_DMAINTEN, TIMERx(x=0..4)
<i>TIMER_DMACFG_DMA_TA_INTF</i>	DMA transfer address is TIMER_INTF, TIMERx(x=0..4)
<i>TIMER_DMACFG_DMA_TA_SWEVG</i>	DMA transfer address is TIMER_SWEVG, TIMERx(x=0..4)
<i>TIMER_DMACFG_DMA_TA_CHCTL0</i>	DMA transfer address is TIMER_CHCTL0, TIMERx(x=0..4)
<i>TIMER_DMACFG_DMA_TA_CHCTL1</i>	DMA transfer address is TIMER_CHCTL1, TIMERx(x=0..4)

<i>TA_CHCTL1</i>	
<i>TIMER_DMACFG_DMA</i> <i>TA_CHCTL2</i>	DMA transfer address is <i>TIMER_CHCTL2</i> , <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_DMACFG_DMA</i> <i>TA_CNT</i>	DMA transfer address is <i>TIMER_CNT</i> , <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_DMACFG_DMA</i> <i>TA_PSC</i>	DMA transfer address is <i>TIMER_PSC</i> , <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_DMACFG_DMA</i> <i>TA_CAR</i>	MA transfer address is <i>TIMER_CAR</i> , <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_DMACFG_DMA</i> <i>TA_CREP</i>	DMA transfer address is <i>TIMER_CREP</i> , <i>TIMERx</i> (<i>x</i> =0,15,16)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH0CV</i>	DMA transfer address is <i>TIMER_CH0CV</i> , <i>TIMERx</i> (<i>x</i> =0..4,15,16)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH1CV</i>	DMA transfer address is <i>TIMER_CH1CV</i> , <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH2CV</i>	DMA transfer address is <i>TIMER_CH2CV</i> , <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH3CV</i>	DMA transfer address is <i>TIMER_CH3CV</i> , <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_DMACFG_DMA</i> <i>TA_CCHP</i>	DMA transfer address is <i>TIMER_CCHP</i> , <i>TIMERx</i> (<i>x</i> =0,15,16)
<i>TIMER_DMACFG_DMA</i> <i>TA_DMACFG</i>	DMA transfer address is <i>TIMER_DMACFG</i> , <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_DMACFG_DMA</i> <i>TA_DMATB</i>	DMA transfer address is <i>TIMER_DMATB</i> , <i>TIMERx</i> (<i>x</i> =0..4)
Input parameter{in}	
<i>dma_lenth</i>	DMA transfer count
<i>TIMER_DMACFG_DMA</i> <i>TC_xTRANSFER</i>	<i>x</i> =1..18, DMA transfer <i>x</i> 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 3-928. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
event	the timer software event generation sources
<i>TIMER_EVENT_SRC_UPG</i>	update event, TIMERx(x=0..5, 15, 16)
<i>TIMER_EVENT_SRC_C0G</i>	channel 0 capture or compare event generation, TIMERx(x=0..4, 15, 16)
<i>TIMER_EVENT_SRC_C1G</i>	channel 1 capture or compare event generation, TIMERx(x=0..4)
<i>TIMER_EVENT_SRC_C2G</i>	channel 2 capture or compare event generation, TIMERx(x=0..4)
<i>TIMER_EVENT_SRC_C3G</i>	channel 3 capture or compare event generation, TIMERx(x=0..4)
<i>TIMER_EVENT_SRC_CMTG</i>	channel commutation event generation, TIMERx(x=0, 15, 16)
<i>TIMER_EVENT_SRC_TRG</i>	trigger event generation, TIMERx(x=0..4)
<i>TIMER_EVENT_SRC_BRK</i>	break event generation, TIMERx(x=0, 15, 16)
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 3-929. 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 3-901. 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 3-930. 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
<i>TIMERx(x=0, 15, 16)</i>	TIMER peripheral selection
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to Table 3-901. 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(TIMERO, &timer_breakpara);

```

timer_break_enable

The description of timer_break_enable is shown as below:

Table 3-931. 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, 15, 16)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable TIMERO break function*/

timer_break_enable (TIMERO);

```

timer_break_disable

The description of timer_break_disable is shown as below:

Table 3-932. 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
<i>TIMERx(x=0, 15, 16)</i>	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 3-933. 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
<i>TIMERx(x=0, 15, 16)</i>	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 3-934. 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
<i>TIMERx(x=0, 15, 16)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 output automatic function */
timer_automatic_output_disable(TIMERO);
```

timer_primary_output_config

The description of timer_primary_output_config is shown as below:

Table 3-935. 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
<i>TIMERx(x=0, 15, 16)</i>	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	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 3-936. 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
<i>TIMERx(x=0, 15, 16)</i>	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	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 3-937. 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, uint32_t ccuctl);
Function descriptions	configure commutation control shadow register update control
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0, 15, 16)</i>	TIMER peripheral selection

Input parameter{in}	
ccuctl	channel control shadow register update control
<i>TIMER_UPDATECTL_CCU</i>	the shadow registers update by when CMTG bit is set
<i>TIMER_UPDATECTL_CUTRI</i>	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 3-938. 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* ocpa);
Function descriptions	initialize the parameters of TIMER channel output parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
ocpa	TIMER channel output parameter struct, the structure members can refer to Table 3-902. 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 3-939. 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* ocpa);
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..4,15,16))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0..4))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0..4))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0..4))
Input parameter{in}	
ocpara	TIMER channel output parameter struct, the structure members can refer to Table 3-902. 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 3-940. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0..4))
Input parameter{in}	
ocmode	channel output compare mode
<i>TIMER_OC_MODE_TIMING</i>	timing mode
<i>TIMER_OC_MODE_ACTIVE</i>	set the channel output
<i>TIMER_OC_MODE_INACTIVE</i>	clear the channel output
<i>TIMER_OC_MODE_TOGGLE</i>	toggle on match
<i>TIMER_OC_MODE_LOW</i>	force low mode
<i>TIMER_OC_MODE_HIGH</i>	force high mode
<i>TIMER_OC_MODE_PWM0</i>	PWM mode 0
<i>TIMER_OC_MODE_PWM1</i>	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 3-941. 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
<i>TIMEx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMEx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMEx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMEx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMEx(x=0..4))
Input parameter{in}	
pulse	channel output pulse value (0~0xFFFFFFFF)
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 3-942. 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

<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0..4))
Input parameter{in}	
ocshadow	channel output shadow state
<i>TIMER_OC_SHADOW_ENABLE</i>	channel output shadow state enable
<i>TIMER_OC_SHADOW_DISABLE</i>	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 3-943. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0..4))

Input parameter{in}	
ocfast	channel output fast function
<i>TIMER_OC_FAST_ENABLE</i>	channel output fast function enable
<i>TIMER_OC_FAST_DISABLE</i>	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 3-944. 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 periph
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0..4))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0..4))
Input parameter{in}	
occlear	channel output clear function
<i>TIMER_OC_CLEAR_ENABLE</i>	channel output clear function enable
<i>TIMER_OC_CLEAR_DISABLE</i>	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 3-945. 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
<i>TIMEx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMEx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMEx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMEx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMEx(x=0..4))
Input parameter{in}	
ocpolarity	channel output polarity
<i>TIMER_OC_POLARITY_HIGH</i>	channel output polarity is high
<i>TIMER_OC_POLARITY_LOW</i>	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 3-946. 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 ocpolarity);
Function descriptions	configure TIMER channel complementary output polarity
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0, 15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0))
Input parameter{in}	
ocpolarity	channel complementary output polarity
<i>TIMER_OCN_POLARITY_HIGH</i>	channel complementary output polarity is high
<i>TIMER_OCN_POLARITY_LOW</i>	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 3-947. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0..4))
Input parameter{in}	
state	TIMER channel enable state
<i>TIMER_CCX_ENABLE</i>	channel enable
<i>TIMER_CCX_DISABLE</i>	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 3-948. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx(x=0,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx(x=0))
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx(x=0))

Input parameter{in}	
state	TIMER channel complementary output enable state
<i>TIMER_CCXN_ENABLE</i>	channel complementary enable
<i>TIMER_CCXN_DISABLE</i>	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 3-949. 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 input parameter struct, the structure members can refer to Table 3-903. 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 3-950. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0..4))
Input parameter{in}	
icpara	TIMER channel input parameter struct, the structure members can refer to Table 3-903. 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_DIRECTTTI;

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 3-951. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0..4))
Input parameter{in}	
prescaler	channel input capture prescaler value
<i>TIMER_IC_PSC_DIV1</i>	no prescaler
<i>TIMER_IC_PSC_DIV2</i>	divided by 2
<i>TIMER_IC_PSC_DIV4</i>	divided by 4
<i>TIMER_IC_PSC_DIV8</i>	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 3-952. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	

channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0..4,15,16))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0..4))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0..4))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0..4))
Output parameter{out}	
-	-
Return value	
uint32_t	channel capture compare register value (0~0xFFFFFFFF)

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 3-953. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
channel	channel to be configured
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
Input parameter{in}	
icpwm	TIMER channel input pwm parameter struct, the structure members can refer to Table 3-903. 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 3-954. Function timer_hall_mode_config

Function name	timer_hall_mode_config
Function prototype	void timer_hall_mode_config(uint32_t timer_periph, uint32_t hallmode);
Function descriptions	configure TIMER hall sensor mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0..4)	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 3-955. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
intrigger	trigger selection
<i>TIMER_SMCFG_TRGS EL_ITI0</i>	Internal trigger input 0
<i>TIMER_SMCFG_TRGS EL_ITI1</i>	Internal trigger input 1
<i>TIMER_SMCFG_TRGS EL_ITI2</i>	Internal trigger input 2
<i>TIMER_SMCFG_TRGS EL_ITI3</i>	Internal trigger input 3
<i>TIMER_SMCFG_TRGS EL_CIOF_ED</i>	CIO edge flag
<i>TIMER_SMCFG_TRGS EL_CIOFE0</i>	channel 0 input Filtered output
<i>TIMER_SMCFG_TRGS EL_C11FE1</i>	channel 1 input Filtered output
<i>TIMER_SMCFG_TRGS EL_ETIFP</i>	External trigger input filter output
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 3-956. 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 outtrigger);
Function descriptions	select TIMER master mode output trigger source

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0..5)</i>	TIMER peripheral selection
Input parameter{in}	
outrigger	master mode control
<i>TIMER_TRI_OUT_SRC_RESET</i>	Reset. When the UPG bit in the <i>TIMERx_SWEVG</i> 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
<i>TIMER_TRI_OUT_SRC_ENABLE</i>	Enable. This mode is useful to start several timers at the same time or to 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.
<i>TIMER_TRI_OUT_SRC_UPDATE</i>	Update. In this mode the master mode controller selects the update event as TRGO.
<i>TIMER_TRI_OUT_SRC_CC0</i>	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.
<i>TIMER_TRI_OUT_SRC_O0CPRE</i>	Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO.
<i>TIMER_TRI_OUT_SRC_O1CPRE</i>	Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO.
<i>TIMER_TRI_OUT_SRC_O2CPRE</i>	Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO.
<i>TIMER_TRI_OUT_SRC_O3CPRE</i>	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 3-957. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
slavemode	slave mode
<i>TIMER_SLAVE_MODE_DISABLE</i>	slave mode disable
<i>TIMER_QUAD_DECODER_MODE0</i>	quadrature decoder mode 0
<i>TIMER_QUAD_DECODER_MODE1</i>	quadrature decoder mode 1
<i>TIMER_QUAD_DECODER_MODE2</i>	quadrature decoder mode 2
<i>TIMER_SLAVE_MODE_RESTART</i>	restart mode
<i>TIMER_SLAVE_MODE_PAUSE</i>	pause mode
<i>TIMER_SLAVE_MODE_EVENT</i>	event mode
<i>TIMER_SLAVE_MODE_EXTERNAL0</i>	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 3-958. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
masterslave	master slave mode state
<i>TIMER_MASTER_SLAVE_MODE_ENABLE</i>	master slave mode enable
<i>TIMER_MASTER_SLAVE_MODE_DISABLE</i>	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 3-959. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
extprescaler	external trigger prescaler
<i>TIMER_EXT_TRI_PSC_OFF</i>	no divided
<i>TIMER_EXT_TRI_PSC_DIV2</i>	divided by 2
<i>TIMER_EXT_TRI_PSC_DIV4</i>	divided by 4

<i>DIV4</i>	
<i>TIMER_EXT_TRI_PSC_DIV8</i>	divided by 8
Input parameter{in}	
expolarity	external trigger polarity
<i>TIMER_ETP_FALLING</i>	active low or falling edge active
<i>TIMER_ETP_RISING</i>	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 3-960. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
decomode	quadrature decoder mode
<i>TIMER_QUAD_DECODER_MODE0</i>	counter counts on CI0FE0 edge depending on CI1FE1 level
<i>TIMER_QUAD_DECODER_MODE1</i>	counter counts on CI1FE1 edge depending on CI0FE0 level
<i>TIMER_QUAD_DECODER_MODE2</i>	counter counts on both CI0FE0 and CI1FE1 edges depending on the level of the other input
Input parameter{in}	
ic0polarity	IC0 polarity

<i>TIMER_IC_POLARITY_RISING</i>	capture rising edge
<i>TIMER_IC_POLARITY_FALLING</i>	capture falling edge
<i>TIMER_IC_POLARITY_BOTH_EDGE</i>	active both edge
Input parameter{in}	
ic1polarity	IC1 polarity
<i>TIMER_IC_POLARITY_RISING</i>	capture rising edge
<i>TIMER_IC_POLARITY_FALLING</i>	capture falling edge
<i>TIMER_IC_POLARITY_BOTH_EDGE</i>	active both 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 3-961. 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
<i>TIMERx(x=0..4)</i>	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 3-962. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
intrigger	trigger selection
<i>TIMER_SMCFG_TRGS_EL_ITI0</i>	Internal trigger input 0 (ITI0)
<i>TIMER_SMCFG_TRGS_EL_ITI1</i>	Internal trigger input 0 (ITI1)
<i>TIMER_SMCFG_TRGS_EL_ITI2</i>	Internal trigger input 0 (ITI2)
<i>TIMER_SMCFG_TRGS_EL_ITI3</i>	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 3-963. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
extrigger	external trigger selection
<i>TIMER_SMCFG_TRGS EL_CIOF_ED</i>	CI0 edge flag (CIOF_ED)
<i>TIMER_SMCFG_TRGS EL_CIOFE0</i>	channel 0 input Filtered output (CIOFE0)
<i>TIMER_SMCFG_TRGS EL_C1FE1</i>	channel 1 input Filtered output (C1FE1)
Input parameter{in}	
expolarity	external trigger polarity
<i>TIMER_IC_POLARITY_ RISING</i>	active high or rising edge active
<i>TIMER_IC_POLARITY_ FALLING</i>	active low or falling edge active
<i>TIMER_IC_POLARITY_ BOTH_EDGE</i>	active both edge
Input parameter{in}	
extfilter	external trigger filter control (0~15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 the external trigger CIOFE0 as external clock input */
```

```
timer_external_trigger_as_external_clock_config(TIMER0,  
TIMER_SMCFG_TRGSEL_CIOFE0, TIMER_IC_POLARITY_RISING, 0);
```

timer_external_clock_mode0_config

The description of timer_external_clock_mode0_config is shown as below:

Table 3-964. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI external trigger prescaler
<i>TIMER_EXT_TRI_PSC_OFF</i>	no divided
<i>TIMER_EXT_TRI_PSC_DIV2</i>	divided by 2
<i>TIMER_EXT_TRI_PSC_DIV4</i>	divided by 4
<i>TIMER_EXT_TRI_PSC_DIV8</i>	divided by 8
Input parameter{in}	
expolarity	ETI external trigger polarity
<i>TIMER_ETP_FALLING</i>	active low or falling edge active
<i>TIMER_ETP_RISING</i>	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 3-965. 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
<i>TIMERx(x=0..4)</i>	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI external trigger prescaler
<i>TIMER_EXT_TRI_PSC_OFF</i>	no divided
<i>TIMER_EXT_TRI_PSC_DIV2</i>	divided by 2
<i>TIMER_EXT_TRI_PSC_DIV4</i>	divided by 4
<i>TIMER_EXT_TRI_PSC_DIV8</i>	divided by 8
Input parameter{in}	
expolarity	ETI external trigger polarity
<i>TIMER_ETP_FALLING</i>	active low or falling edge active
<i>TIMER_ETP_RISING</i>	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 3-966. 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
<i>TIMERx(x=0..4)</i>	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 3-967. 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, 15, 16)	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 3-968. 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
<i>TIMERx (x=0,15,16)</i>	TIMER peripheral selection
Input parameter{in}	
outsel	output value selection
<i>TIMER_OUTSEL_DISABLE</i>	no effect
<i>TIMER_OUTSEL_ENABLE</i>	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_flag_get

The description of timer_flag_get is shown as below:

Table 3-969. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
<i>TIMER_FLAG_UP</i>	update flag, TIMERx(x=0..5,15,16)
<i>TIMER_FLAG_CH0</i>	channel 0 flag, TIMERx(x=0..4,15,16)
<i>TIMER_FLAG_CH1</i>	channel 1 flag, TIMERx(x=0..4)
<i>TIMER_FLAG_CH2</i>	channel 2 flag, TIMERx(x=0..4)
<i>TIMER_FLAG_CH3</i>	channel 3 flag, TIMERx(x=0..4)
<i>TIMER_FLAG_CMT</i>	channel commutation flag, TIMERx(x=0,15,16)
<i>TIMER_FLAG_TRG</i>	trigger flag, TIMERx(x=0..4)

<i>TIMER_FLAG_BRK</i>	break flag, <i>TIMERx</i> (<i>x</i> =0,15,16)
<i>TIMER_FLAG_CH0O</i>	channel 0 overcapture flag, <i>TIMERx</i> (<i>x</i> =0..4,15,16)
<i>TIMER_FLAG_CH1O</i>	channel 1 overcapture flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_FLAG_CH2O</i>	channel 2 overcapture flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_FLAG_CH3O</i>	channel 3 overcapture flag, <i>TIMERx</i> (<i>x</i> =0..4)
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 3-970. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
<i>TIMER_FLAG_UP</i>	update flag, <i>TIMERx</i> (<i>x</i> =0..5,15,16)
<i>TIMER_FLAG_CH0</i>	channel 0 flag, <i>TIMERx</i> (<i>x</i> =0..4,15,16)
<i>TIMER_FLAG_CH1</i>	channel 1 flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_FLAG_CH2</i>	channel 2 flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_FLAG_CH3</i>	channel 3 flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_FLAG_CMT</i>	channel commutation flag, <i>TIMERx</i> (<i>x</i> =0,15,16)
<i>TIMER_FLAG_TRG</i>	trigger flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_FLAG_BRK</i>	break flag, <i>TIMERx</i> (<i>x</i> =0,15,16)
<i>TIMER_FLAG_CH0O</i>	channel 0 overcapture flag, <i>TIMERx</i> (<i>x</i> =0..4,15,16)
<i>TIMER_FLAG_CH1O</i>	channel 1 overcapture flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_FLAG_CH2O</i>	channel 2 overcapture flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_FLAG_CH3O</i>	channel 3 overcapture flag, <i>TIMERx</i> (<i>x</i> =0..4)
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* clear TIMER0 update flags */
```

```
timer_flag_clear(TIMER0, TIMER_FLAG_UP);
```

timer_interrupt_enable

The description of timer_interrupt_enable is shown as below:

Table 3-971. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
interrupt	timer interrupt enable source
<i>TIMER_INT_UP</i>	update interrupt enable, TIMERx (x=0..5,15,16)
<i>TIMER_INT_CH0</i>	channel 0 interrupt enable, TIMERx (x=0..4,15,16)
<i>TIMER_INT_CH1</i>	channel 1 interrupt enable, TIMERx (x=0..4)
<i>TIMER_INT_CH2</i>	channel 2 interrupt enable, TIMERx (x=0..4)
<i>TIMER_INT_CH3</i>	channel 3 interrupt enable, TIMERx (x=0..4)
<i>TIMER_INT_CMT</i>	commutation interrupt enable, TIMERx (x=0,15,16)
<i>TIMER_INT_TRG</i>	trigger interrupt enable, TIMERx (x=0..4)
<i>TIMER_INT_BRK</i>	break interrupt enable, TIMERx (x=0,15,16)
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 3-972. 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..5,15,16)
TIMER_INT_CH0	channel 0 interrupt disable, TIMERx(x=0..4,15,16)
TIMER_INT_CH1	channel 1 interrupt disable, TIMERx(x=0..4)
TIMER_INT_CH2	channel 2 interrupt disable, TIMERx(x=0..4)
TIMER_INT_CH3	channel 3 interrupt disable, TIMERx(x=0..4)
TIMER_INT_CMT	commutation interrupt disable, TIMERx(x=0,15,16)
TIMER_INT_TRG	trigger interrupt disable, TIMERx(x=0..4)
TIMER_INT_BRK	break interrupt disable, TIMERx(x=0,15,16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMERO0 update interrupt */
```

```
timer_interrupt_disable(TIMERO0, TIMER_INT_UP);
```

timer_interrupt_flag_get

The description of timer_interrupt_flag_get is shown as below:

Table 3-973. Function timer_interrupt_flag_get

Function name	timer_interrupt_flag_get
Function prototype	FlagStatus timer_interrupt_flag_get(uint32_t timer_periph, uint32_t int_flag);
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}	

int_flag	the timer interrupt bits
<i>TIMER_INT_FLAG_UP</i>	update interrupt flag, <i>TIMERx</i> (<i>x</i> =0..5,15,16)
<i>TIMER_INT_FLAG_CH0</i>	channel 0 interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4,15,16)
<i>TIMER_INT_FLAG_CH1</i>	channel 1 interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_INT_FLAG_CH2</i>	channel 2 interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_INT_FLAG_CH3</i>	channel 3 interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_INT_FLAG_CMT</i>	channel commutation interrupt flag, <i>TIMERx</i> (<i>x</i> =0,15,16)
<i>TIMER_INT_FLAG_TRG</i>	trigger interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_INT_FLAG_BRK</i>	break interrupt flag, <i>TIMERx</i> (<i>x</i> =0,15,16)
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 3-974. Function timer_interrupt_flag_clear

Function name	timer_interrupt_flag_clear
Function prototype	void timer_interrupt_flag_clear(uint32_t timer_periph, uint32_t int_flag);
Function descriptions	clear TIMER interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
int_flag	the timer interrupt bits
<i>TIMER_INT_FLAG_UP</i>	update interrupt flag, <i>TIMERx</i> (<i>x</i> =0..5,15,16)
<i>TIMER_INT_FLAG_CH0</i>	channel 0 interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4,15,16)
<i>TIMER_INT_FLAG_CH1</i>	channel 1 interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_INT_FLAG_CH2</i>	channel 2 interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_INT_FLAG_CH3</i>	channel 3 interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4)
<i>TIMER_INT_FLAG_CMT</i> <i>T</i>	channel commutation interrupt flag, <i>TIMERx</i> (<i>x</i> =0,15,16)
<i>TIMER_INT_FLAG_TRG</i>	trigger interrupt flag, <i>TIMERx</i> (<i>x</i> =0..4)

<i>TIMER_INT_FLAG_BRK</i>	break interrupt flag, <i>TIMERx</i> (<i>x</i> =0,15,16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TIMER0 update interrupt flag */
```

```
timer_interrupt_flag_clear(TIMER0, TIMER_INT_FLAG_UP);
```

3.28. TRNG

The true random number generator (RNG) module can generate a 32-bit value using continuous analog noise. The TRNG registers are listed in chapter [3.28.1](#). the TRNG firmware functions are introduced in chapter [3.28.2](#).

3.28.1. Descriptions of Peripheral registers

TRNG registers are listed in the table shown as below:

Table 3-975 TRNG Registers

Registers	Descriptions
TRNG_CTL	TRNG control register
TRNG_STAT	TRNG status register
TRNG_DATA	TRNG data register

3.28.2. Descriptions of Peripheral functions

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

Table 3-976. TRNG firmware function

Function name	Function description
trng_deinit	deinitialize the TRNG
trng_enable	enable the TRNG interface
trng_disable	disable the TRNG interface
trng_get_true_random_data	get the true random data
trng_interrupt_enable	enable the TRNG interrupt
trng_interrupt_disable	disable the TRNG interrupt
trng_flag_get	get the TRNG status flags
trng_interrupt_flag_get	get the TRNG interrupt flags
trng_interrupt_flag_clear	clear the TRNG interrupt flags

Enum trng_flag_enum

Table 3-977. Enum trng_flag_enum

Member name	Function description
TRNG_FLAG_DRDY	random data ready status
TRNG_FLAG_CECS	clock error current status
TRNG_FLAG_SECS	seed error current status

Enum trng_int_flag_enum

Table 3-978. Enum trng_int_flag_enum

Member name	Function description
TRNG_INT_FLAG_CEIF	clock error interrupt flag
TRNG_INT_FLAG_SEIF	seed error interrupt flag

trng_deinit

The description of trng_deinit is shown as below:

Table 3-979. Function trng_deinit

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

Example:

```
/* reset TRNG */
trng_deinit();
```

trng_enable

The description of trng_enable is shown as below:

Table 3-980. Function trng_enable

Function name	trng_enable
Function prototype	void trng_enable(void)
Function descriptions	enable the TRNG interface
Precondition	-

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

Example:

```
/* enable TRNG */
```

```
trng_enable();
```

trng_disable

The description of trng_disable is shown as below:

Table 3-981 Function trng_disable

Function name	trng_disable
Function prototype	void trng_disable(void);
Function descriptions	disable the TRNG interface
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TRNG */
```

```
trng_disable();
```

trng_get_true_random_data

The description of trng_get_true_random_data is shown as below:

Table 3-982 Function trng_get_true_random_data

Function name	trng_get_true_random_data
Function prototype	uint32_t trng_get_true_random_data(void)
Function descriptions	get the true random data
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the generated random data

Example:

```
/* get true random data */
uint32_t data;
data = trng_get_true_random_data();
```

trng_interrupt_enable

The description of trng_interrupt_enable is shown as below:

Table 3-983 trng_interrupt_enable

Function name	trng_interrupt_enable
Function prototype	void trng_interrupt_enable(void);
Function descriptions	enable the TRNG interrupt
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TRNG interrupt */
trng_interrupt_enable();
```

trng_interrupt_disable

The description of trng_interrupt_disable is shown as below:

Table 3-984 trng_interrupt_disable

Function name	trng_interrupt_disable
Function prototype	void trng_interrupt_disable(void);
Function descriptions	disable the TRNG interrupt
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* disable TRNG interrupt */
```

```
trng_interrupt_disable();
```

trng_flag_get

The description of trng_flag_get is shown as below:

Table 3-985 trng_flag_get

Function name	trng_flag_get
Function prototype	FlagStatus trng_flag_get(trng_flag_enum flag);
Function descriptions	get the trng status flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	trng_flag_enum, please refer to Table 3-977. Enum trng_flag_enum
TRNG_FLAG_DRDY	random data ready status
TRNG_FLAG_CECS	clock error current status
TRNG_FLAG_SECS	seed error current status
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TRNG clock error current flag status */
```

```
FlagStatus flag_status = RESET;
```

```
flag_status == trng_flag_get(TRNG_FLAG_CECS);
```

trng_interrupt_flag_get

The description of trng_interrupt_flag_get is shown as below:

Table 3-986 trng_interrupt_flag_get

Function name	trng_interrupt_flag_get
Function prototype	FlagStatus trng_interrupt_flag_get(trng_int_flag_enum int_flag);
Function descriptions	get the trng interrupt flags

Precondition	-
The called functions	-
Input parameter{in}	
flag	trng_flag_enum, please refer to Table 3-978. Enum trng_int_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TRNG clock error interrupt flag */
FlagStatus interrupt_flag = RESET;
interrupt_flag = trng_interrupt_flag_get(TRNG_INT_FLAG_CEIF);
```

trng_interrupt_flag_clear

The description of trng_interrupt_flag_clear is shown as below:

Table 3-987 trng_interrupt_flag_clear

Function name	trng_interrupt_flag_clear
Function prototype	void trng_interrupt_flag_get(trng_int_flag_enum int_flag);
Function descriptions	clear the trng interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	trng_flag_enum, please refer to Table 3-978. Enum trng_int_flag_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TRNG clock error interrupt flag */
trng_interrupt_flag_clear(TRNG_INT_FLAG_CEIF);
```

3.29. TSI

Touch Sensing Interface (TSI) provides a convenient solution for touch keys, sliders and capacitive proximity sensing applications. The TSI registers are listed in chapter [3.29.1](#), the TSI firmware functions are introduced in chapter [3.29.2](#).

3.29.1. Descriptions of Peripheral registers

TSI registers are listed in the table shown as below:

Table 3-988. TSI Registers

Registers	Descriptions
TSI_CTL0	control register0
TSI_INTEN	interrupt enable register
TSI_INTC	interrupt flag clear register
TSI_INTF	interrupt flag register
TSI_PHM	pin hysteresis mode register
TSI_ASW	analog switch register
TSI_SAMPCFG	sample configuration register
TSI_CHCFG	channel configuration register
TSI_GCTL	group control register
TSI_GxCYCN (x=0..2)	group x cycle number registers
TSI_CTL1	control register1

3.29.2. Descriptions of Peripheral functions

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

Table 3-989. TSI firmware function

Function name	Function description
tsi_deinit	reset TSI peripheral
tsi_init	initialize TSI plus prescaler,charge plus,transfer plus,max cycle number
tsi_enable	enable TSI module
tsi_disable	disable TSI module
tsi_sample_pin_enable	enable sample pin
tsi_sample_pin_disable	disable sample pin
tsi_channel_pin_enable	enable channel pin
tsi_channel_pin_disable	disable channel pin
tsi_software_mode_config	configure TSI triggering by software
tsi_software_start	start a charge-transfer sequence when TSI is in software trigger mode
tsi_software_stop	stop a charge-transfer sequence when TSI is in software trigger mode
tsi_hardware_mode_config	configure TSI triggering by hardware
tsi_pin_mode_config	configure TSI pin mode when charge-transfer sequence is IDLE
tsi_extend_charge_config	configure extend charge state
tsi_plus_config	configure charge plus and transfer plus

Function name	Function description
tsi_max_number_config	configure the max cycle number of a charge-transfer sequence
tsi_hysteresis_on	switch on hysteresis pin
tsi_hysteresis_off	switch off hysteresis pin
tsi_analog_on	switch on analog pin
tsi_analog_off	switch off analog pin
tsi_interrupt_enable	enable TSI interrupt
tsi_interrupt_disable	disable TSI interrupt
tsi_interrupt_flag_clear	Clear TSI interrupt flag
tsi_interrupt_flag_get	get TSI interrupt flag
tsi_flag_clear	clear flag
tsi_flag_get	get flag
tsi_group_enable	enable group
tsi_group_disable	disable group
tsi_group_status_get	get group complete status
tsi_group0_cycle_get	get the cycle number for group0 as soon as a charge-transfer sequence completes
tsi_group1_cycle_get	get the cycle number for group1 as soon as a charge-transfer sequence completes
tsi_group2_cycle_get	get the cycle number for group2 as soon as a charge-transfer sequence completes

tsi_deinit

The description of tsi_deinit is shown as below:

Table 3-990. Function tsi_deinit

Function name	tsi_deinit
Function prototype	void tsi_deinit(void);
Function descriptions	reset TSI peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset TSI*/
tsi_deinit();
```

tsi_init

The description of tsi_init is shown as below:

Table 3-991. Function tsi_init

Function name	tsi_init
Function prototype	void tsi_init(uint32_t prescaler,uint32_t charge_duration,uint32_t transfer_duration,uint32_t max_number);
Function descriptions	initialize TSI plus prescaler,charge plus,transfer plus,max cycle number
Precondition	-
The called functions	-
Input parameter{in}	
prescaler	CTCLK clock division factor
<i>TSI_CTCDIV_DIV1</i>	$f_{CTCLK} = f_{HCLK}$
<i>TSI_CTCDIV_DIV2</i>	$f_{CTCLK} = f_{HCLK}/2$
<i>TSI_CTCDIV_DIV4</i>	$f_{CTCLK} = f_{HCLK}/4$
<i>TSI_CTCDIV_DIV8</i>	$f_{CTCLK} = f_{HCLK}/8$
<i>TSI_CTCDIV_DIV16</i>	$f_{CTCLK} = f_{HCLK}/16$
<i>TSI_CTCDIV_DIV32</i>	$f_{CTCLK} = f_{HCLK}/32$
<i>TSI_CTCDIV_DIV64</i>	$f_{CTCLK} = f_{HCLK}/64$
<i>TSI_CTCDIV_DIV128</i>	$f_{CTCLK} = f_{HCLK}/128$
<i>TSI_CTCDIV_DIV256</i>	$f_{CTCLK} = f_{HCLK}/256$
<i>TSI_CTCDIV_DIV512</i>	$f_{CTCLK} = f_{HCLK}/512$
<i>TSI_CTCDIV_DIV1024</i>	$f_{CTCLK} = f_{HCLK}/1024$
<i>TSI_CTCDIV_DIV2048</i>	$f_{CTCLK} = f_{HCLK}/2048$
<i>TSI_CTCDIV_DIV4096</i>	$f_{CTCLK} = f_{HCLK}/4096$
<i>TSI_CTCDIV_DIV8192</i>	$f_{CTCLK} = f_{HCLK}/8192$
<i>TSI_CTCDIV_DIV16384</i> 4	$f_{CTCLK} = f_{HCLK}/16384$
<i>TSI_CTCDIV_DIV32768</i> 8	$f_{CTCLK} = f_{HCLK}/32768$
Input parameter{in}	
charge_duration	charge state duration time
<i>TSI_CHARGE_1CTCLK(x=1..16)</i>	the duration time of charge state is x CTCLK
Input parameter{in}	
transfer_duration	charge transfer state duration time
<i>TSI_TRANSFER_xCTCLK(x=1..16)SS_CLEAR</i>	the duration time of transfer state is x CTCLK
Input parameter{in}	
max_number	max cycle number
<i>TSI_MAXNUM255</i>	the max cycle number of a sequence is 255
<i>TSI_MAXNUM511</i>	the max cycle number of a sequence is 511

<i>TSI_MAXNUM1023</i>	the max cycle number of a sequence is 1023
<i>TSI_MAXNUM2047</i>	the max cycle number of a sequence is 2047
<i>TSI_MAXNUM4095</i>	the max cycle number of a sequence is 4095
<i>TSI_MAXNUM8191</i>	the max cycle number of a sequence is 8191
<i>TSI_MAXNUM16383</i>	the max cycle number of a sequence is 16383
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* init TSI*/
```

```
tsi_init (TSI_CTCDIV_DIV4096, TSI_CHARGE_10CTCLK, TSI_TRANSFER_8CTCLK,
TSI_MAXNUM511);
```

tsi_enable

The description of tsi_enable is shown as below:

Table 3-992. Function tsi_enable

Function name	tsi_enable
Function prototype	void tsi_enable (void);
Function descriptions	enable TSI module
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TSI module */
```

```
tsi_enable();
```

tsi_disable

The description of tsi_disable is shown as below:

Table 3-993. Function tsi_disable

Function name	tsi_disable
Function prototype	void tsi_disable (void);
Function descriptions	disable TSI module

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

Example:

```
/* disable TSI module */
```

```
tsi_disable ();
```

tsi_sample_pin_enable

The description of tsi_sample_pin_enable is shown as below:

Table 3-994. Function tsi_sample_pin_enable

Function name	tsi_sample_pin_enable
Function prototype	void tsi_sample_pin_enable(uint32_t sample);
Function descriptions	enable sample pin
Precondition	-
The called functions	-
Input parameter{in}	
sample	sample pin
<i>TSI_SAMPCFG_GxPy(x=0..2,y=0..3)</i>	pin y of group x is sample pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable G1P2 sample pin */
```

```
tsi_sample_pin_enable (TSI_SAMPCFG_G1P2);
```

tsi_sample_pin_disable

The description of tsi_sample_pin_disable is shown as below:

Table 3-995. Function tsi_sample_pin_disable

Function name	tsi_sample_pin_disable
Function prototype	void tsi_sample_pin_disable(uint32_t sample);
Function descriptions	disable sample pin

Precondition	-
The called functions	-
Input parameter{in}	
sample	sample pin
<i>TSI_SAMPCFG_GxPy</i> (<i>x=0..2,y=0..3</i>)	pin y of group x is sample pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable G1P2 sample pin */
tsi_sample_pin_disable (TSI_SAMPCFG_G1P2);
```

tsi_channel_pin_enable

The description of tsi_channel_pin_enable is shown as below:

Table 3-996. Function tsi_channel_pin_enable

Function name	tsi_channel_pin_enable
Function prototype	void tsi_channel_pin_enable(uint32_t channel);
Function descriptions	enable channel pin
Precondition	-
The called functions	-
Input parameter{in}	
channel	channel pin
<i>TSI_CHCFG_GxPy</i> (<i>x=</i> <i>0..2,y=0..3</i>)	pin y of group x
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable G1P2 channel pin */
tsi_channel_pin_enable (TSI_CHCFG_G1P2);
```

tsi_channel_pin_disable

The description of tsi_channel_pin_disable is shown as below:

Table 3-997. Function tsi_channel_pin_disable

Function name	tsi_channel_pin_disable
----------------------	-------------------------

Function prototype	void tsi_channel_pin_disable(uint32_t channel);
Function descriptions	disable channel pin
Precondition	-
The called functions	-
Input parameter{in}	
channel	channel pin
<i>TSI_CHCFG_GxPy(x=0..2,y=0..3)</i>	pin y of group x
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable G1P2 channel pin */
tsi_channel_pin_disable (TSI_CHCFG_G1P2);
```

tsi_sofeware_mode_config

The description of tsi_sofeware_mode_config is shown as below:

Table 3-998. Function tsi_software_mode_config

Function name	tsi_software_mode_config
Function prototype	void tsi_software_mode_config (void);
Function descriptions	configure TSI triggering by software
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TSI triggering by software */
tsi_software_mode_config ();
```

tsi_software_start

The description of tsi_software_start is shown as below:

Table 3-999. Function tsi_software_start

Function name	tsi_software_start
----------------------	--------------------

Function prototype	void tsi_software_start (void);
Function descriptions	start a charge-transfer sequence when TSI is in software trigger mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start a charge-transfer sequence when TSI is in software trigger mode */
tsi_software_start ();
```

tsi_software_stop

The description of tsi_software_stop is shown as below:

Table 3-1000. Function tsi_software_stop

Function name	tsi_software_stop
Function prototype	void tsi_software_stop (void);
Function descriptions	stop a charge-transfer sequence when TSI is in software trigger mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop a charge-transfer sequence when TSI is in software trigger mode */
tsi_software_stop ();
```

tsi_hardware_mode_config

The description of tsi_hardware_mode_config is shown as below:

Table 3-1001. Function tsi_hardware_mode_config

Function name	tsi_hardware_mode_config
Function prototype	void tsi_hardware_mode_config(uint8_t trigger_edge);
Function descriptions	configure TSI triggering by hardware

Precondition	-
The called functions	-
Input parameter{in}	
trigger_edge	the edge type in hardware trigger mode
<i>TSI_FALLING_TRIGGER</i>	falling edge trigger TSI charge transfer sequence
<i>TSI_RISING_TRIGGER</i>	rising edge trigger TSI charge transfer sequence
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TSI triggering by hardware */
```

```
tsi_hardware_mode_config (TSI_FALLING_TRIGGER);
```

tsi_pin_mode_config

The description of tsi_pin_mode_config is shown as below:

Table 3-1002. Function tsi_pin_mode_config

Function name	tsi_pin_mode_config
Function prototype	void tsi_pin_mode_config(uint8_t pin_mode);
Function descriptions	configure TSI pin mode when charge-transfer sequence is IDLE
Precondition	-
The called functions	-
Input parameter{in}	
pin_mode	pin mode when charge-transfer sequence is IDLE
<i>TSI_OUTPUT_LOW</i>	TSI pin will output low when IDLE
<i>TSI_INPUT_FLOATING</i>	TSI pin will keep input_floating when IDLE
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TSI pin mode when charge-transfer sequence is IDLE */
```

```
tsi_pin_mode_config (TSI_OUTPUT_LOW);
```

tsi_extend_charge_config

The description of tsi_extend_charge_config is shown as below:

Table 3-1003. Function tsi_extend_charge_config

Function name	tsi_extend_charge_config
Function prototype	void tsi_extend_charge_config(ControlStatus extend,uint8_t prescaler,uint32_t max_duration);
Function descriptions	configure extend charge state
Precondition	-
The called functions	-
Input parameter{in}	
extend	enable or disable extend charge state
<i>ENABLE</i>	enable extend charge state
<i>DISABLE</i>	disable extend charge state
Input parameter{in}	
prescaler	ECCLK clock division factor
<i>TSI_EXTEND_DIV1</i>	$f_{ECCLK} = f_{HCLK}$
<i>TSI_EXTEND_DIV2</i>	$f_{ECCLK} = f_{HCLK} / 2$
<i>TSI_EXTEND_DIV3</i>	$f_{ECCLK} = f_{HCLK} / 3$
<i>TSI_EXTEND_DIV4</i>	$f_{ECCLK} = f_{HCLK} / 4$
<i>TSI_EXTEND_DIV5</i>	$f_{ECCLK} = f_{HCLK} / 5$
<i>TSI_EXTEND_DIV6</i>	$f_{ECCLK} = f_{HCLK} / 6$
<i>TSI_EXTEND_DIV7</i>	$f_{ECCLK} = f_{HCLK} / 7$
<i>TSI_EXTEND_DIV8</i>	$f_{ECCLK} = f_{HCLK} / 8$
Input parameter{in}	
max_duration	extend charge state maximum duration time
<i>value range 1...128</i>	extend charge state maximum duration time is $1 * t_{ECCLK} \sim 128 * t_{ECCLK}$
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure extend charge state */
```

```
tsi_extend_charge_config (ENABLE, TSI_EXTEND_DIV2, 10);
```

tsi_plus_config

The description of tsi_plus_config is shown as below:

Table 3-1004. Function tsi_plus_config

Function name	tsi_plus_config
Function prototype	void tsi_plus_config(uint32_t prescaler,uint32_t charge_duration,uint32_t transfer_duration);
Function descriptions	configure charge plus and transfer plus
Precondition	-

The called functions	-
Input parameter{in}	
prescaler	CTCLK clock division factor
<i>TSI_CTCDIV_DIV1</i>	$f_{CTCLK} = f_{HCLK}$
<i>TSI_CTCDIV_DIV2</i>	$f_{CTCLK} = f_{HCLK} / 2$
<i>TSI_CTCDIV_DIV4</i>	$f_{CTCLK} = f_{HCLK} / 4$
<i>TSI_CTCDIV_DIV8</i>	$f_{CTCLK} = f_{HCLK} / 8$
<i>TSI_CTCDIV_DIV16</i>	$f_{CTCLK} = f_{HCLK} / 16$
<i>TSI_CTCDIV_DIV32</i>	$f_{CTCLK} = f_{HCLK} / 32$
<i>TSI_CTCDIV_DIV64</i>	$f_{CTCLK} = f_{HCLK} / 64$
<i>TSI_CTCDIV_DIV128</i>	$f_{CTCLK} = f_{HCLK} / 128$
<i>TSI_CTCDIV_DIV256</i>	$f_{CTCLK} = f_{HCLK} / 256$
<i>TSI_CTCDIV_DIV512</i>	$f_{CTCLK} = f_{HCLK} / 512$
<i>TSI_CTCDIV_DIV1024</i>	$f_{CTCLK} = f_{HCLK} / 1024$
<i>TSI_CTCDIV_DIV2048</i>	$f_{CTCLK} = f_{HCLK} / 2048$
<i>TSI_CTCDIV_DIV4096</i>	$f_{CTCLK} = f_{HCLK} / 4096$
<i>TSI_CTCDIV_DIV8192</i>	$f_{CTCLK} = f_{HCLK} / 8192$
<i>TSI_CTCDIV_DIV16384</i> 4	$f_{CTCLK} = f_{HCLK} / 16384$
<i>TSI_CTCDIV_DIV32768</i> 8	$f_{CTCLK} = f_{HCLK} / 32768$
Input parameter{in}	
charge_duration	charge state duration time
<i>TSI_CHARGE_1CTCL</i> <i>K(x=1..16)</i>	the duration time of charge state is x CTCLK
Input parameter{in}	
transfer_duration	charge transfer state duration time
<i>TSI_TRANSFER_xCTC</i> <i>LK(x=1..16)</i>	the duration time of transfer state is x CTCLK
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure charge plus and transfer plus */
```

```
tsi_plus_config (TSI_CTCDIV_DIV1024, TSI_CHARGE_10CTCLK,  
TSI_TRANSFER_8CTCLK);
```

tsi_max_number_config

The description of tsi_max_number_config is shown as below:

Table 3-1005. Function tsi_max_number_config

Function name	tsi_max_number_config
Function prototype	void tsi_max_number_config(uint32_t max_number);
Function descriptions	configure the max cycle number of a charge-transfer sequence
Precondition	-
The called functions	-
Input parameter{in}	
max_number	max cycle number
<i>TSI_MAXNUM255</i>	the max cycle number of a sequence is 255
<i>TSI_MAXNUM511</i>	the max cycle number of a sequence is 511
<i>TSI_MAXNUM1023</i>	the max cycle number of a sequence is 1023
<i>TSI_MAXNUM2047</i>	the max cycle number of a sequence is 2047
<i>TSI_MAXNUM4095</i>	the max cycle number of a sequence is 4095
<i>TSI_MAXNUM8191</i>	the max cycle number of a sequence is 8191
<i>TSI_MAXNUM16383</i>	the max cycle number of a sequence is 16383
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the max cycle number of a charge-transfer sequence */
```

```
tsi_max_number_config (TSI_MAXNUM1023);
```

tsi_hysteresis_on

The description of tsi_hysteresis_on is shown as below:

Table 3-1006. Function tsi_hysteresis_on

Function name	tsi_hysteresis_on
Function prototype	void tsi_hysteresis_on(uint32_t group_pin);
Function descriptions	switch on hysteresis pin
Precondition	-
The called functions	-
Input parameter{in}	
group_pin	select pin which will be switched on hysteresis
<i>TSI_PHM_GxPy(x=0..2, y=0..3)</i>	pin y of group x switch on hysteresis
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* switch on hysteresis pin */
```

```
tsi_hysteresis_on (TSI_PHM_G1P2);
```

tsi_hysteresis_off

The description of tsi_hysteresis_off is shown as below:

Table 3-1007. Function tsi_hysteresis_off

Function name	tsi_hysteresis_off
Function prototype	void tsi_hysteresis_off(uint32_t group_pin);
Function descriptions	switch off hysteresis pin
Precondition	-
The called functions	-
Input parameter{in}	
group_pin	select pin which will be switched off hysteresis
<i>TSI_PHM_GxPy</i> (<i>x</i> =0..2, <i>y</i> =0..3)	pin <i>y</i> of group <i>x</i> switch off hysteresis
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* switch off hysteresis pin */
```

```
tsi_hysteresis_off (TSI_PHM_G1P2);
```

tsi_analog_on

The description of tsi_analog_on is shown as below:

Table 3-1008. Function tsi_analog_on

Function name	tsi_analog_on
Function prototype	void tsi_analog_on(uint32_t group_pin);
Function descriptions	switch on analog pin
Precondition	-
The called functions	-
Input parameter{in}	
group_pin	select pin which will be switched on analog
<i>TSI_ASW_GxPy</i> (<i>x</i> =0..2, <i>y</i> =0..3)	pin <i>y</i> of group <i>x</i> switch on analog
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* switch on analog pin */
```

```
tsi_analog_on (TSI_ASW_G1P2);
```

tsi_analog_off

The description of tsi_analog_off is shown as below:

Table 3-1009. Function tsi_analog_off

Function name	tsi_analog_off
Function prototype	void tsi_analog_off(uint32_t group_pin);
Function descriptions	switch off analog pin
Precondition	-
The called functions	-
Input parameter{in}	
group_pin	select pin which will be switched off analog
<i>TSI_ASW_GxPy</i> (x=0..2,y=0..3)	pin y of group x switch off analog
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* switch off analog pin */
```

```
tsi_analog_off (TSI_ASW_G1P2);
```

tsi_group_enable

The description of tsi_group_enable is shown as below:

Table 3-1010. Function tsi_group_enable

Function name	tsi_group_enable
Function prototype	void tsi_group_enable(uint32_t group);
Function descriptions	enable group
Precondition	-
The called functions	-
Input parameter{in}	
group	select group to be enabled
<i>TSI_GCTL_GEx</i> (x=0..2)	the x group will be enabled
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable group 2 */
```

```
tsi_group_enable (TSI_GCTL_GE2);
```

tsi_group_disable

The description of tsi_group_disable is shown as below:

Table 3-1011. Function tsi_group_disable

Function name	tsi_group_disable
Function prototype	void tsi_group_disable(uint32_t group);
Function descriptions	disbale group
Precondition	-
The called functions	-
Input parameter{in}	
group	select group to be disabled
<i>TSI_GCTL_GEx(x=0..2)</i>	the x group will be disabled
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable group 2 */
```

```
tsi_group_disable (TSI_GCTL_GE2);
```

tsi_group_status_get

The description of tsi_group_status_get is shown as below:

Table 3-1012. Function tsi_group_status_get

Function name	tsi_group_status_get
Function prototype	FlagStatus tsi_group_status_get(uint32_t group);
Function descriptions	get group complete status
Precondition	-
The called functions	-
Input parameter{in}	
group	select group
<i>TSI_GCTL_GCx(x=0..2)</i>	get the complete status of group x
Output parameter{out}	
-	-
Return value	

FlagStatus	SET or RESET
-------------------	--------------

Example:

```
/* get group complete status */
```

```
FlagStatus flag = RESET;
```

```
flag = tsi_flag_get (TSI_GCTL_GC2);
```

tsi_group0_cycle_get

The description of tsi_group0_cycle_get is shown as below:

Table 3-1013. Function tsi_group0_cycle_get

Function name	tsi_group0_cycle_get
Function prototype	uint16_t tsi_group0_cycle_get(void);
Function descriptions	get the cycle number for group0 as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0-8192

Example:

```
/* get the cycle number for group0 as soon as a charge-transfer sequence completes */
```

```
uint16_t flag = 0;
```

```
flag = tsi_group0_cycle_get ();
```

tsi_group1_cycle_get

The description of tsi_group1_cycle_get is shown as below:

Table 3-1014. Function tsi_group1_cycle_get

Function name	tsi_group1_cycle_get
Function prototype	uint16_t tsi_group1_cycle_get(void);
Function descriptions	get the cycle number for group1 as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
uint16_t	0-8192

Example:

```
/* get the cycle number for group1 as soon as a charge-transfer sequence completes */
```

```
uint16_t flag = 0;
```

```
flag = tsi_group1_cycle_get ();
```

tsi_group2_cycle_get

The description of tsi_group2_cycle_get is shown as below:

Table 3-1015. Function tsi_group2_cycle_get

Function name	tsi_group2_cycle_get
Function prototype	uint16_t tsi_group2_cycle_get(void);
Function descriptions	get the cycle number for group2 as soon as a charge-transfer sequence completes
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	0-8192

Example:

```
/* get the cycle number for group2 as soon as a charge-transfer sequence completes */
```

```
uint16_t flag = 0;
```

```
flag = tsi_group2_cycle_get ();
```

tsi_flag_clear

The description of tsi_flag_clear is shown as below:

Table 3-1016. Function tsi_flag_clear

Function name	tsi_flag_clear
Function prototype	void tsi_flag_clear(uint32_t flag);
Function descriptions	clear flag
Precondition	-
The called functions	-

Input parameter{in}	
flag	select flag which will be cleared
<i>TSI_FLAG_CTCF</i>	clear charge-transfer complete flag
<i>TSI_FLAG_MNERR</i>	clear max cycle number error
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TSI_FLAG_CTCF_CLR flag */
tsi_flag_clear (TSI_FLAG_CTCF_CLR);
```

tsi_flag_get

The description of tsi_flag_get is shown as below:

Table 3-1017. Function tsi_flag_get

Function name	tsi_flag_get
Function prototype	FlagStatus tsi_flag_get(uint32_t flag);
Function descriptions	get flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
<i>TSI_FLAG_CTCF</i>	charge-transfer complete flag
<i>TSI_FLAG_MNERR</i>	max Cycle Number Error
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TSI_FLAG_CTCF flag */
FlagStatus flag = RESET;
flag = tsi_flag_get (TSI_FLAG_CTCF);
```

tsi_interrupt_enable

The description of tsi_interrupt_enable is shown as below:

Table 3-1018. Function tsi_interrupt_enable

Function name	tsi_interrupt_enable
----------------------	----------------------

Function prototype	void tsi_interrupt_enable(uint32_t source);
Function descriptions	enable TSI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
source	select interrupt which will be enabled
<i>TSI_INT_CCTCF</i>	charge-transfer complete flag interrupt enable
<i>TSI_INT_MNERR</i>	max cycle number error interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TSI_INT_CCTCF interrupt */
tsi_interrupt_enable (TSI_INT_CCTCF);
```

tsi_interrupt_disable

The description of tsi_interrupt_disable is shown as below:

Table 3-1019. Function tsi_interrupt_disable

Function name	tsi_interrupt_disable
Function prototype	void tsi_interrupt_disable(uint32_t source);
Function descriptions	disable TSI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
source	select interrupt which will be disabled
<i>TSI_INT_CCTCF</i>	charge-transfer complete flag interrupt disable
<i>TSI_INT_MNERR</i>	max cycle number error interrupt disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TSI_INT_CCTCF interrupt */
tsi_interrupt_disable (TSI_INT_CCTCF);
```

tsi_interrupt_flag_clear

The description of tsi_interrupt_flag_clear is shown as below:

Table 3-1020. Function tsi_interrupt_flag_clear

Function name	tsi_interrupt_flag_clear
Function prototype	void tsi_interrupt_flag_clear(uint32_t flag);
Function descriptions	clear TSI interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	select flag which will be cleared
TSI_INT_FLAG_CTCF	clear charge-transfer complete flag
TSI_INT_FLAG_MNER R	clear max cycle number error
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TSI_INT_FLAG_CTCF interrupt flag */
tsi_interrupt_flag_clear (TSI_INT_FLAG_CTCF);
```

tsi_interrupt_flag_get

The description of tsi_interrupt_flag_get is shown as below:

Table 3-1021. Function tsi_interrupt_flag_get

Function name	tsi_interrupt_flag_get
Function prototype	FlagStatus tsi_interrupt_flag_get(uint32_t flag);
Function descriptions	get TSI interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag
TSI_INT_FLAG_CTCF	charge-transfer complete flag
TSI_INT_FLAG_MNER R	max Cycle Number Error
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TSI_INT_FLAG_CTCF interrupt flag */
FlagStatus flag = RESET;
```

```
flag = tsi_interrupt_flag_get (TSI_INT_FLAG_CTCF);
```

3.30. TZPCU

TZPCU is used to implement TrustZone protection functions for the devices. The TZPCU registers are listed in chapter [3.30.1](#), the TZPCU firmware functions are introduced in chapter [3.30.2](#).

3.30.1. Descriptions of Peripheral registers

TZPCU registers are listed in the table shown as below:

Table 3-1022. TZPCU Registers

Registers	Descriptions
TZSPC registers	
TZPCU_TZSPC_CTL	TZSPC control register
TZPCU_TZSPC_SAM_CFG0	TZSPC secure access mode configuration register 0
TZPCU_TZSPC_SAM_CFG1	TZSPC secure access mode configuration register 1
TZPCU_TZSPC_SAM_CFG2	TZSPC secure access mode configuration register 2
TZPCU_TZSPC_PAM_CFG0	TZSPC privilege access mode configuration register 0
TZPCU_TZSPC_PAM_CFG1	TZSPC privilege access mode configuration register 1
TZPCU_TZSPC_PAM_CFG2	TZSPC privilege access mode configuration register 2
TZPCU_TZSPC_TZMMPC0_NSM0	TZSPC external memory 0 non-secure mark register 0
TZPCU_TZSPC_TZMMPC0_NSM1	TZSPC external memory 0 non-secure mark register 1
TZPCU_TZSPC_TZMMPC0_NSM2	TZSPC external memory 0 non-secure mark register 2
TZPCU_TZSPC_TZMMPC0_NSM3	TZSPC external memory 0 non-secure mark register 3
TZPCU_TZSPC_TZMMPC1_NSM0	TZSPC external memory 1 non-secure mark register 0
TZPCU_TZSPC_TZMMPC1_NSM1	TZSPC external memory 1 non-secure mark register 1
TZPCU_TZSPC_DBG_CFG	TZSPC debug configuration register
TZBMPC registers	
TZPCU_TZBMPC_CTL	TZBMPC control register
TZPCU_TZBMPC_VEC	TZBMPC vector register
TZPCU_TZBMPC_LOCK0	TZBMPC lock register 0
TZIAC registers	
TZPCU_TZIAC_INTEN0	TZIAC enable register 0
TZPCU_TZIAC_INTEN1	TZIAC enable register 1
TZPCU_TZIAC_INTEN2	TZIAC enable register 2
TZPCU_TZIAC_STAT0	TZIAC status register 0
TZPCU_TZIAC_STAT1	TZIAC status register 1
TZPCU_TZIAC_STAT2	TZIAC status register 2
TZPCU_TZIAC_STATC0	TZIAC flag clear register 0
TZPCU_TZIAC_STATC1	TZIAC flag clear register 1
TZPCU_TZIAC_STATC2	TZIAC flag clear register 2

3.30.2. Descriptions of Peripheral functions

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

Table 3-1023. TZPCU firmware function

Function name	Function description
TZSPC function	
tzpcu_tzspc_peripheral_attributes_config	configure peripherals secure attributes
tzpcu_tzspc_peripheral_attributes_get	get peripherals secure attributes
tzpcu_non_secure_mark_struct_parameter_init	initialize the TZPCU non-secure mark struct with the default values
tzpcu_tzspc_emnsm_config	configure external memory non-secure mark
tzpcu_tzspc_items_lock	lock TZSPC items
tzpcu_tzspc_dbg_config	configure debug type
TZBMPC function	
tzpcu_tzbmpc_lock	lock the control register of the TZBMPC sub-block
tzpcu_tzbmpc_security_state_config	configure default security state
tzpcu_tzbmpc_secure_access_config	configure secure access state
tzpcu_tzbmpc_block_secure_access_mode_config	configure block secure access mode
tzpcu_tzbmpc_union_block_lock	lock configure union block secure access mode
TZIAC function	
tzpcu_tziac_interrupt_enable	enable illegal access interrupt
tzpcu_tziac_interrupt_disable	disable illegal access interrupt
tzpcu_tziac_flag_get	get illegal access interrupt flag
tzpcu_tziac_flag_clear	clear illegal access interrupt flag

Enum tzpcu_mem

Table 3-1024. tzpcu_mem

enum name	enum description
QSPI_FLASH_MEM	QSPI flash memory
QSPI_PSRAM_MEM	QSPI PSRAM memory

Enum tzpcu_non_secure_mark_region

Table 3-1025. tzpcu_non_secure_mark_region

enum name	enum description
NON_SECURE_MARK_REGION0	non-secure mark region 0
NON_SECURE_MARK_REGION1	non-secure mark region 1
NON_SECURE_MARK_REGION2	non-secure mark region 2 (QSPI flash only)
NON_SECURE_MARK_REGION3	non-secure mark region 3 (QSPI flash only)

Structure tzpcu_non_secure_mark_struct

Table 3-1026. tzpcu_non_secure_mark_struct

Member name	Function description
memory_type	memory type, refer to Table 3-1024. tzpcu_mem
region_number	non secure mark region number, refer to Table 3-1025. tzpcu_non_secure_mark_region
start_address	external memory non-secure area start address
length	external memory non-secure area length

tzpcu_tzspc_peripheral_attributes_config

The description of tzpcu_tzspc_peripheral_attributes_config is shown as below:

Table 3-1027. Function tzpcu_tzspc_peripheral_attributes_config

Function name	tzpcu_tzspc_peripheral_attributes_config
Function prototype	void tzpcu_tzspc_peripheral_attributes_config(uint32_t periph, uint32_t attributes);
Function descriptions	configure peripherals secure attributes
Precondition	-
The called functions	-
Input parameter{in}	
periph	peripheral
TZPCU_PERIPH_TIME R1	TIMER1 peripheral
TZPCU_PERIPH_TIME R2	TIMER2 peripheral
TZPCU_PERIPH_TIME R3	TIMER3 peripheral
TZPCU_PERIPH_TIME R4	TIMER4 peripheral
TZPCU_PERIPH_TIME R5	TIMER5 peripheral
TZPCU_PERIPH_WW DGT	WWDGT peripheral
TZPCU_PERIPH_FWD GT	FWDGT peripheral
TZPCU_PERIPH_SPI1	SPI1 peripheral
TZPCU_PERIPH_USA RT1	USART1 peripheral
TZPCU_PERIPH_USA RT2	USART2 peripheral
TZPCU_PERIPH_I2C0	I2C0 peripheral
TZPCU_PERIPH_I2C1	I2C1 peripheral

TZPCU_PERIPH_USB FS	USBFS peripheral
TZPCU_PERIPH_TIME R0	TIMER0 peripheral
TZPCU_PERIPH_SPI0	SPI0 peripheral
TZPCU_PERIPH_USA RT0	USART0 peripheral
TZPCU_PERIPH_TIME R15	TIMER15 peripheral
TZPCU_PERIPH_TIME R16	TIMER16 peripheral
TZPCU_PERIPH_HPD F	HPDF peripheral(HPDF not support on GD32W515Tx series devices)
TZPCU_PERIPH_CRC	CRC peripheral
TZPCU_PERIPH_TSI	TSI peripheral
TZPCU_PERIPH_ICAC HE	ICACHE peripheral
TZPCU_PERIPH_ADC	ADC peripheral
TZPCU_PERIPH_CAU	CAU peripheral
TZPCU_PERIPH_HAU	HAU peripheral
TZPCU_PERIPH_TRN G	TRNG peripheral
TZPCU_PERIPH_PKC AU	PKCAU peripheral
TZPCU_PERIPH_SDIO	SDIO peripheral
TZPCU_PERIPH_EFU SE	EFUSE peripheral
TZPCU_PERIPH_DBG	DBG peripheral(only for privilege attributes)
TZPCU_PERIPH_SQPI _PSRAMREG	SQPI PSRAMREG peripheral
TZPCU_PERIPH_QSPI _FLASHREG	QSPI FLASHREG peripheral
TZPCU_PERIPH_WIFI _RF	WIFI RF peripheral
TZPCU_PERIPH_I2S1 _ADD	I2S1_ADD peripheral
TZPCU_PERIPH_DCI	DCI peripheral(DCI not support on GD32W515Tx series devices)
TZPCU_PERIPH_WIFI	WIFI peripheral
TZPCU_PERIPH_ALL	all peripherals
Input parameter{in}	
attributes	security and privilege attributes
TZPCU_SEC	peripheral secure attributes is secure, exclusive with TZPCU_NSEC
TZPCU_NSEC	peripheral secure attributes is non-secure, exclusive with TZPCU_SEC

<i>TZPCU_PRIV</i>	peripheral privilege attributes is privilege, exclusive with TZPCU_NPRIV
<i>TZPCU_NPRIV</i>	peripheral privilege attributes is non-privilege, exclusive with TZPCU_PRIV
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER1 secure attributes to secure and non-privilege */
```

```
tzpcu_tzspc_peripheral_attributes_config(TZPCU_PERIPH_TIMER1, TZPCU_SEC |  
TZPCU_NPRIV);
```

tzpcu_tzspc_peripheral_attributes_get

The description of tzpcu_tzspc_peripheral_attributes_get is shown as below:

Table 3-1028. Function tzpcu_tzspc_peripheral_attributes_get

Function name	tzpcu_tzspc_peripheral_attributes_get
Function prototype	uint32_t tzpcu_tzspc_peripheral_attributes_get(uint32_t periph);
Function descriptions	get peripherals secure attributes
Precondition	-
The called functions	-
Input parameter{in}	
periph	peripheral
<i>TZPCU_PERIPH_TIMER1</i>	TIMER1 peripheral
<i>TZPCU_PERIPH_TIMER2</i>	TIMER2 peripheral
<i>TZPCU_PERIPH_TIMER3</i>	TIMER3 peripheral
<i>TZPCU_PERIPH_TIMER4</i>	TIMER4 peripheral
<i>TZPCU_PERIPH_TIMER5</i>	TIMER5 peripheral
<i>TZPCU_PERIPH_WWDGT</i>	WWDGT peripheral
<i>TZPCU_PERIPH_FWDGT</i>	FWDGT peripheral
<i>TZPCU_PERIPH_SPI1</i>	SPI1 peripheral
<i>TZPCU_PERIPH_USART1</i>	USART1 peripheral
<i>TZPCU_PERIPH_USART2</i>	USART2 peripheral

TZPCU_PERIPH_I2C0	I2C0 peripheral
TZPCU_PERIPH_I2C1	I2C1 peripheral
TZPCU_PERIPH_USB FS	USBFS peripheral
TZPCU_PERIPH_TIME R0	TIMER0 peripheral
TZPCU_PERIPH_SPI0	SPI0 peripheral
TZPCU_PERIPH_USA RT0	USART0 peripheral
TZPCU_PERIPH_TIME R15	TIMER15 peripheral
TZPCU_PERIPH_TIME R16	TIMER16 peripheral
TZPCU_PERIPH_HPD F	HPDF peripheral(c)
TZPCU_PERIPH_CRC	CRC peripheral
TZPCU_PERIPH_TSI	TSI peripheral
TZPCU_PERIPH_ICAC HE	ICACHE peripheral
TZPCU_PERIPH_ADC	ADC peripheral
TZPCU_PERIPH_CAU	CAU peripheral
TZPCU_PERIPH_HAU	HAU peripheral
TZPCU_PERIPH_TRN G	TRNG peripheral
TZPCU_PERIPH_PKC AU	PKCAU peripheral
TZPCU_PERIPH_SDIO	SDIO peripheral
TZPCU_PERIPH_EFU SE	EFUSE peripheral
TZPCU_PERIPH_DBG	DBG peripheral(only for privilege attributes)
TZPCU_PERIPH_SQPI _PSRAMREG	SQPI PSRAMREG peripheral
TZPCU_PERIPH_QSPI _FLASHREG	QSPI FLASHREG peripheral
TZPCU_PERIPH_WIFI _RF	WIFI RF peripheral
TZPCU_PERIPH_I2S1 _ADD	I2S1_ADD peripheral
TZPCU_PERIPH_DCI	DCI peripheral(DCI not support on GD32W515Tx series devices)
TZPCU_PERIPH_WIFI	WIFI peripheral
TZPCU_PERIPH_ALL	all peripherals
Output parameter{out}	
-	-

Return value	
uint32_t	TZPCU_SEC/TZPCU_NSEC or TZPCU_PRIV/TZPCU_NPRIV

Example:

```
/* get TIMER1 secure attributes */
```

```
uint32_t sec_attr = tzpcu_tzspc_peripheral_attributes_get(TZPCU_PERIPH_TIMER1);
```

tzpcu_non_secure_mark_struct_para_init

The description of tzpcu_non_secure_mark_struct_para_init is shown as below:

Table 3-1029. Function tzpcu_non_secure_mark_struct_para_init

Function name	tzpcu_non_secure_mark_struct_para_init
Function prototype	void tzpcu_non_secure_mark_struct_para_init(tzpcu_non_secure_mark_struct* init_struct);
Function descriptions	initialize the TZPCU non-secure mark struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	TZPCU non-secure mark init parameter struct, the structure members can refer to Table 3-1026. tzpcu_non_secure_mark_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the TZPCU non-secure mark struct with the default values */
```

```
tzpcu_non_secure_mark_struct init_struct;
```

```
tzpcu_non_secure_mark_struct_para_init(&init_struct);
```

tzpcu_tzspc_emnsm_config

The description of tzpcu_tzspc_emnsm_config is shown as below:

Table 3-1030. Function tzpcu_tzspc_emnsm_config

Function name	tzpcu_tzspc_emnsm_config
Function prototype	ErrStatus tzpcu_tzspc_emnsm_config(tzpcu_non_secure_mark_struct *p_non_secure_mark);
Function descriptions	configure external memory non-secure mark
Precondition	-
The called functions	-

Input parameter{in}	
p_non_secure_mark	TZPCU non-secure mark struct, the structure members can refer to Table 3-1026. tzpcu_non_secure_mark_struct .
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```

/* configure external memory QSPI FLASH first 8KB space to non-secure */

tzpcu_non_secure_mark_struct non_secure_mark;

ErrStatus status = ERROR;

tzpcu_non_secure_mark_struct_para_init(&non_secure_mark);

non_secure_mark.memory_type = QSPI_FLASH_MEM;

non_secure_mark.region_number = NON_SECURE_MARK_REGION0;

non_secure_mark.start_address = 0U;

non_secure_mark.length = 0x00002000;

status = tzpcu_tzspc_emnsm_config(&non_secure_mark);

```

tzpcu_tzspc_items_lock

The description of tzpcu_tzspc_items_lock is shown as below:

Table 3-1031. Function tzpcu_tzspc_items_lock

Function name	tzpcu_tzspc_items_lock
Function prototype	void tzpcu_tzspc_items_lock(void);
Function descriptions	lock tzspc items
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* lock tzspc items */

tzpcu_tzspc_items_lock();

```

tzpcu_tzspsc_dbg_config

The description of tzpcu_tzspsc_dbg_config is shown as below:

Table 3-1032. Function tzpcu_tzspsc_dbg_config

Function name	tzpcu_tzspsc_dbg_config
Function prototype	void tzpcu_tzspsc_dbg_config(uint32_t dbg_type, ControlStatus config_value);
Function descriptions	configure debug type
Precondition	-
The called functions	-
Input parameter{in}	
dbg_type	debug type
INVASIVE_DEBUG	invasive debug
NON_INVASIVE_DEBUG	non-invasive debug
SECURE_INVASIVE_DEBUG	secure invasive debug
SECURE_NON_INVASIVE_DEBUG	secure non-Invasive debug
Input parameter{in}	
config_value	ENABLE or DISABLE
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable invasive debug */
```

```
tzpcu_tzspsc_dbg_config(INVASIVE_DEBUG, DISABLE);
```

tzpcu_tzbmpc_lock

The description of tzpcu_tzbmpc_lock is shown as below:

Table 3-1033. Function tzpcu_tzbmpc_lock

Function name	tzpcu_tzbmpc_lock
Function prototype	void tzpcu_tzbmpc_lock(uint32_t tzbmpx);
Function descriptions	lock the control register of the TZBMPC sub-block
Precondition	-
The called functions	-
Input parameter{in}	
tzbmp	TZBMPC
TZBMPCx	TZBMPCx(x=0,1,2,3)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the control register of the TZBMPC0 sub-block */
```

```
tzpcu_tzbmpc_lock(TZBMPC0);
```

tzpcu_tzbmpc_security_state_config

The description of tzpcu_tzbmpc_security_state_config is shown as below:

Table 3-1034. Function tzpcu_tzbmpc_security_state_config

Function name	tzpcu_tzbmpc_security_state_config
Function prototype	void tzpcu_tzbmpc_security_state_config(uint32_t tzbmpx, uint32_t sec_state);
Function descriptions	configure default security state
Precondition	-
The called functions	-
Input parameter{in}	
tzbmp	TZBMPC
<i>TZBMPCx</i>	TZBMPCx(x=0,1,2,3)
Input parameter{in}	
sec_state	security state of TZBMPC source clock
<i>DEFAULT_STATE</i>	default state, if do not exist secure area, source clock is non-secure
<i>INVERT_STATE</i>	invert the state, even if do not exist secure area, source clock is still secure
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TZBMPC0 default security state to invert state */
```

```
tzpcu_tzbmpc_security_state_config(TZBMPC0, INVERT_STATE);
```

tzpcu_tzbmpc_secure_access_config

The description of tzpcu_tzbmpc_secure_access_config is shown as below:

Table 3-1035. Function tzpcu_tzbmpc_secure_access_config

Function name	tzpcu_tzbmpc_secure_access_config
Function prototype	void tzpcu_tzbmpc_secure_access_config(uint32_t tzbmpx, uint32_t sec_illaccess_state);

Function descriptions	configure secure access state
Precondition	-
The called functions	-
Input parameter{in}	
tzbmp	TZBMPC
<i>TZBMPCx</i>	TZBMPCx(x=0,1,2,3)
Input parameter{in}	
sec_illaccess_state	secure illegal access state
<i>SECURE_ILLEGAL_ACCESS_ENABLE</i>	secure read/write access non-secure SRAM is illegal
<i>SECURE_ILLEGAL_ACCESS_DISABLE</i>	secure read/write access non-secure SRAM is legal
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TZBMPC0 secure access state to be enabled , secure read/write access non-secure SRAM is illegal */
```

```
tzpcu_tzbmpc_secure_access_config(TZBMPC0, SECURE_ILLEGAL_ACCESS_ENABLE);
```

tzpcu_tzbmpc_block_secure_access_mode_config

The description of tzpcu_tzbmpc_block_secure_access_mode_config is shown as below:

Table 3-1036. Function tzpcu_tzbmpc_block_secure_access_mode_config

Function name	tzpcu_tzbmpc_block_secure_access_mode_config
Function prototype	void tzpcu_tzbmpc_block_secure_access_mode_config(uint32_t tzbmpx, uint32_t block_pos_num, uint32_t access_mode);
Function descriptions	configure block secure access mode
Precondition	-
The called functions	-
Input parameter{in}	
tzbmp	TZBMPC
<i>TZBMPCx</i>	TZBMPCx(x=0,1,2,3)
Input parameter{in}	
block_pos_num	block position number(for TZBMPC0 and TZBMPC1 block position number is 0-255; for TZBMPC2 block position number is 0-511; for TZBMPC3 block position number is 0-799)
Input parameter{in}	
access_mode	bolck secure access mode

<i>BLOCK_SECURE_ACCESS_MODE_SEC</i>	block secure access mode is secure
<i>BLOCK_SECURE_ACCESS_MODE_NSEC</i>	block secure access mode is non-secure
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TZBMPC0 block 0 secure access mode to secure */
```

```
tzpcu_tzbmpc_block_secure_access_mode_config(TZBMPC0, 0, BLOCK_SECURE_ACCESS_MODE_SEC);
```

tzpcu_tzbmpc_union_block_lock

The description of tzpcu_tzbmpc_union_block_lock is shown as below:

Table 3-1037. Function tzpcu_tzbmpc_union_block_lock

Function name	tzpcu_tzbmpc_union_block_lock
Function prototype	void tzpcu_tzbmpc_union_block_lock(uint32_t tzbmpx, uint32_t union_block_position_num);
Function descriptions	lock configuration union block secure access mode
Precondition	-
The called functions	-
Input parameter{in}	
tzbmp	TZBMPC
<i>TZBMPCx</i>	TZBMPCx(x=0,1,2,3)
Input parameter{in}	
union_block_position_num	union block position number (for TZBMPC0 and TZBMPC1 union block position number is 0-7; for TZBMPC2 union block position number is 0-15; for TZBMPC3 union block position number is 0-23)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock configure union block secure access mode of TZBMPC0 union block 0 */
```

```
tzpcu_tzbmpc_union_block_lock(TZBMPC0, 0);
```

tzpcu_tziac_interrupt_enable

The description of tzpcu_tziac_interrupt_enable is shown as below:

Table 3-1038. Function tzpcu_tziac_interrupt_enable

Function name	tzpcu_tziac_interrupt_enable
Function prototype	void tzpcu_tziac_interrupt_enable(uint32_t periph);
Function descriptions	enable illegal access interrupt
Precondition	-
The called functions	-
Input parameter{in}	
periph	peripheral
TZPCU_PERIPH_TIME R1	TIMER1 peripheral
TZPCU_PERIPH_TIME R2	TIMER2 peripheral
TZPCU_PERIPH_TIME R3	TIMER3 peripheral
TZPCU_PERIPH_TIME R4	TIMER4 peripheral
TZPCU_PERIPH_TIME R5	TIMER5 peripheral
TZPCU_PERIPH_WW DGT	WWDGT peripheral
TZPCU_PERIPH_FWD GT	FWDGT peripheral
TZPCU_PERIPH_SPI1	SPI1 peripheral
TZPCU_PERIPH_USA RT1	USART1 peripheral
TZPCU_PERIPH_USA RT2	USART2 peripheral
TZPCU_PERIPH_I2C0	I2C0 peripheral
TZPCU_PERIPH_I2C1	I2C1 peripheral
TZPCU_PERIPH_USB FS	USBFS peripheral
TZPCU_PERIPH_TIME R0	TIMER0 peripheral
TZPCU_PERIPH_SPI0	SPI0 peripheral
TZPCU_PERIPH_USA RT0	USART0 peripheral
TZPCU_PERIPH_TIME R15	TIMER15 peripheral
TZPCU_PERIPH_TIME R16	TIMER16 peripheral

<i>TZPCU_PERIPH_HPDI</i>	HPDF peripheral(HPDF not support on GD32W515Tx series devices)
<i>TZPCU_PERIPH_CRC</i>	CRC peripheral
<i>TZPCU_PERIPH_TSI</i>	TSI peripheral
<i>TZPCU_PERIPH_ICACHE</i>	ICACHE peripheral
<i>TZPCU_PERIPH_ADC</i>	ADC peripheral
<i>TZPCU_PERIPH_CAU</i>	CAU peripheral
<i>TZPCU_PERIPH_HAU</i>	HAU peripheral
<i>TZPCU_PERIPH_TRNG</i>	TRNG peripheral
<i>TZPCU_PERIPH_PKCAU</i>	PKCAU peripheral
<i>TZPCU_PERIPH_SDIO</i>	SDIO peripheral
<i>TZPCU_PERIPH_RTC</i>	RTC peripheral
<i>TZPCU_PERIPH_PMU</i>	PMU peripheral
<i>TZPCU_PERIPH_SYSCFG</i>	SYSCFG peripheral
<i>TZPCU_PERIPH_DMA0</i>	DMA0 peripheral
<i>TZPCU_PERIPH_DMA1</i>	DMA1 peripheral
<i>TZPCU_PERIPH_RCU</i>	RCU peripheral
<i>TZPCU_PERIPH_FLASH</i>	FLASH peripheral
<i>TZPCU_PERIPH_FMC</i>	FLASH REG peripheral
<i>TZPCU_PERIPH_EXTI</i>	EXTI peripheral
<i>TZPCU_PERIPH_TZSPC</i>	TZSPC peripheral
<i>TZPCU_PERIPH_TZIAC</i>	TZIAC peripheral
<i>TZPCU_PERIPH_SRAM0</i>	SRAM0 peripheral
<i>TZPCU_PERIPH_TZBMPC0_REG</i>	ZBMPC0 register
<i>TZPCU_PERIPH_SRAM1</i>	SRAM1 peripheral
<i>TZPCU_PERIPH_TZBMPC1_REG</i>	TZBMPC1 register
<i>TZPCU_PERIPH_SRAM2</i>	SRAM2 peripheral
<i>TZPCU_PERIPH_TZBMPC2_REG</i>	TZBMPC2 register

<code>TZPCU_PERIPH_SRAM3</code>	SRAM3 peripheral
<code>TZPCU_PERIPH_TZBMPC3_REG</code>	TZBMPC3 register
<code>TZPCU_PERIPH_EFUSE</code>	EFUSE peripheral
<code>TZPCU_PERIPH_SQPI_PSRAM</code>	SQPI_PSRAM peripheral
<code>TZPCU_PERIPH_QSPI_FLASH</code>	QSPI_FLASH peripheral
<code>TZPCU_PERIPH_SQPI_PSRAMREG</code>	SQPI PSRAMREG peripheral
<code>TZPCU_PERIPH_QSPI_FLASHREG</code>	QSPI FLASHREG peripheral
<code>TZPCU_PERIPH_WIFI_RF</code>	WIFI RF peripheral
<code>TZPCU_PERIPH_I2S1_ADD</code>	I2S1_ADD peripheral
<code>TZPCU_PERIPH_DCI</code>	DCI peripheral(DCI not support on GD32W515Tx series devices)
<code>TZPCU_PERIPH_WIFI</code>	WIFI peripheral
<code>TZPCU_PERIPH_ALL</code>	all peripherals
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER1 illegal access interrupt */
```

```
tzpcu_tziac_interrupt_enable(TZPCU_PERIPH_TIMER1);
```

tzpcu_tziac_interrupt_disable

The description of `tzpcu_tziac_interrupt_disable` is shown as below:

Table 3-1039. Function `tzpcu_tziac_interrupt_disable`

Function name	<code>tzpcu_tziac_interrupt_disable</code>
Function prototype	<code>void tzpcu_tziac_interrupt_disable (uint32_t periph);</code>
Function descriptions	disable illegal access interrupt
Precondition	-
The called functions	-
Input parameter{in}	
periph	peripheral
<code>TZPCU_PERIPH_TIMER1</code>	TIMER1 peripheral

<i>R1</i>	
<i>TZPCU_PERIPH_TIME</i> <i>R2</i>	TIMER2 peripheral
<i>TZPCU_PERIPH_TIME</i> <i>R3</i>	TIMER3 peripheral
<i>TZPCU_PERIPH_TIME</i> <i>R4</i>	TIMER4 peripheral
<i>TZPCU_PERIPH_TIME</i> <i>R5</i>	TIMER5 peripheral
<i>TZPCU_PERIPH_WW</i> <i>DGT</i>	WWDGT peripheral
<i>TZPCU_PERIPH_FWD</i> <i>GT</i>	FWDGT peripheral
<i>TZPCU_PERIPH_SPI1</i>	SPI1 peripheral
<i>TZPCU_PERIPH_USA</i> <i>RT1</i>	USART1 peripheral
<i>TZPCU_PERIPH_USA</i> <i>RT2</i>	USART2 peripheral
<i>TZPCU_PERIPH_I2C0</i>	I2C0 peripheral
<i>TZPCU_PERIPH_I2C1</i>	I2C1 peripheral
<i>TZPCU_PERIPH_USB</i> <i>FS</i>	USBFS peripheral
<i>TZPCU_PERIPH_TIME</i> <i>R0</i>	TIMER0 peripheral
<i>TZPCU_PERIPH_SPI0</i>	SPI0 peripheral
<i>TZPCU_PERIPH_USA</i> <i>RT0</i>	USART0 peripheral
<i>TZPCU_PERIPH_TIME</i> <i>R15</i>	TIMER15 peripheral
<i>TZPCU_PERIPH_TIME</i> <i>R16</i>	TIMER16 peripheral
<i>TZPCU_PERIPH_HPD</i> <i>F</i>	HPDF peripheral(HPDF not support on GD32W515Tx series devices)
<i>TZPCU_PERIPH_CRC</i>	CRC peripheral
<i>TZPCU_PERIPH_TSI</i>	TSI peripheral
<i>TZPCU_PERIPH_ICAC</i> <i>HE</i>	ICACHE peripheral
<i>TZPCU_PERIPH_ADC</i>	ADC peripheral
<i>TZPCU_PERIPH_CAU</i>	CAU peripheral
<i>TZPCU_PERIPH_HAU</i>	HAU peripheral
<i>TZPCU_PERIPH_TRN</i> <i>G</i>	TRNG peripheral
<i>TZPCU_PERIPH_PKC</i>	PKCAU peripheral

AU	
TZPCU_PERIPH_SDIO	SDIO peripheral
TZPCU_PERIPH_RTC	RTC peripheral
TZPCU_PERIPH_PMU	PMU peripheral
TZPCU_PERIPH_SYS CFG	SYSCFG peripheral
TZPCU_PERIPH_DMA 0	DMA0 peripheral
TZPCU_PERIPH_DMA 1	DMA1 peripheral
TZPCU_PERIPH_RCU	RCU peripheral
TZPCU_PERIPH_FLAS H	FLASH peripheral
TZPCU_PERIPH_FMC	FLASH REG peripheral
TZPCU_PERIPH_EXTI	EXTI peripheral
TZPCU_PERIPH_TZS PC	TZSPC peripheral
TZPCU_PERIPH_TZIA C	TZIAC peripheral
TZPCU_PERIPH_SRA M0	SRAM0 peripheral
TZPCU_PERIPH_TZB MPC0_REG	ZBMPC0 register
TZPCU_PERIPH_SRA M1	SRAM1 peripheral
TZPCU_PERIPH_TZB MPC1_REG	TZBMPC1 register
TZPCU_PERIPH_SRA M2	SRAM2 peripheral
TZPCU_PERIPH_TZB MPC2_REG	TZBMPC2 register
TZPCU_PERIPH_SRA M3	SRAM3 peripheral
TZPCU_PERIPH_TZB MPC3_REG	TZBMPC3 register
TZPCU_PERIPH_EFU SE	EFUSE peripheral
TZPCU_PERIPH_SQPI _PSRAM	SQPI_PSRAM peripheral
TZPCU_PERIPH_QSPI _FLASH	QSPI_FLASH peripheral
TZPCU_PERIPH_SQPI _PSRAMREG	SQPI PSRAMREG peripheral

<i>TZPCU_PERIPH_QSPI_FLASHREG</i>	QSPI FLASHREG peripheral
<i>TZPCU_PERIPH_WIFI_RF</i>	WIFI RF peripheral
<i>TZPCU_PERIPH_I2S1_ADD</i>	I2S1_ADD peripheral
<i>TZPCU_PERIPH_DCI</i>	DCI peripheral(DCI not support on GD32W515Tx series devices)
<i>TZPCU_PERIPH_WIFI</i>	WIFI peripheral
<i>TZPCU_PERIPH_ALL</i>	all peripherals
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER1 llegal access interrupt */
```

```
tzpcu_tziac_interrupt_disable(TZPCU_PERIPH_TIMER1);
```

tzpcu_tziac_flag_get

The description of `tzpcu_tziac_flag_get` is shown as below:

Table 3-1040. Function `tzpcu_tziac_flag_get`

Function name	<code>tzpcu_tziac_flag_get</code>
Function prototype	<code>uint32_t tzpcu_tziac_flag_get (uint32_t periph_flag);</code>
Function descriptions	get llegal access interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
periph_flag	peripheral interrupt flag
<i>TZPCU_TZIAC_FLAG_TIMER1</i>	TIMER1 peripheral
<i>TZPCU_TZIAC_FLAG_TIMER2</i>	TIMER2 peripheral
<i>TZPCU_TZIAC_FLAG_TIMER3</i>	TIMER3 peripheral
<i>TZPCU_TZIAC_FLAG_TIMER4</i>	TIMER4 peripheral
<i>TZPCU_TZIAC_FLAG_TIMER5</i>	TIMER5 peripheral
<i>TZPCU_TZIAC_FLAG_WWDGT</i>	WWDGT peripheral
<i>TZPCU_TZIAC_FLAG_FWDGT</i>	FWDGT peripheral

<i>FWDGT</i>	
<i>TZPCU_TZIAC_FLAG_</i> <i>SPI1</i>	SPI1 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>USART1</i>	USART1 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>USART2</i>	USART2 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>I2C0</i>	I2C0 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>I2C1</i>	I2C1 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>USBFS</i>	USBFS peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TIMER0</i>	TIMER0 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>SPI0</i>	SPI0 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>USART0</i>	USART0 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TIMER15</i>	TIMER15 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TIMER16</i>	TIMER16 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>HPDF</i>	HPDF peripheral(HPDF not support on GD32W515Tx series devices)
<i>TZPCU_TZIAC_FLAG_</i> <i>CRC</i>	CRC peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TSI</i>	TSI peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>ICACHE</i>	ICACHE peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>ADC</i>	ADC peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>CAU</i>	CAU peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>HAU</i>	HAU peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TRNG</i>	TRNG peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>PKCAU</i>	PKCAU peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>SDIO</i>	SDIO peripheral

<i>TZPCU_TZIAC_FLAG_RTC</i>	RTC peripheral
<i>TZPCU_TZIAC_FLAG_PMU</i>	PMU peripheral
<i>TZPCU_TZIAC_FLAG_SYSCFG</i>	SYSCFG peripheral
<i>TZPCU_TZIAC_FLAG_DMA0</i>	DMA0 peripheral
<i>TZPCU_TZIAC_FLAG_DMA1</i>	DMA1 peripheral
<i>TZPCU_TZIAC_FLAG_RCU</i>	RCU peripheral
<i>TZPCU_TZIAC_FLAG_FLASH</i>	FLASH peripheral
<i>TZPCU_TZIAC_FLAG_FMC</i>	FLASH REG peripheral
<i>TZPCU_TZIAC_FLAG_EXTI</i>	EXTI peripheral
<i>TZPCU_TZIAC_FLAG_TZSPC</i>	TZSPC peripheral
<i>TZPCU_TZIAC_FLAG_TZIAC</i>	TZIAC peripheral
<i>TZPCU_TZIAC_FLAG_SRAM0</i>	SRAM0 peripheral
<i>TZPCU_TZIAC_FLAG_TZBMPC0_REG</i>	ZBMPC0 register
<i>TZPCU_TZIAC_FLAG_SRAM1</i>	SRAM1 peripheral
<i>TZPCU_TZIAC_FLAG_TZBMPC1_REG</i>	TZBMPC1 register
<i>TZPCU_TZIAC_FLAG_SRAM2</i>	SRAM2 peripheral
<i>TZPCU_TZIAC_FLAG_TZBMPC2_REG</i>	TZBMPC2 register
<i>TZPCU_TZIAC_FLAG_SRAM3</i>	SRAM3 peripheral
<i>TZPCU_TZIAC_FLAG_TZBMPC3_REG</i>	TZBMPC3 register
<i>TZPCU_TZIAC_FLAG_EFUSE</i>	EFUSE peripheral
<i>TZPCU_TZIAC_FLAG_SQPI_PSRAM</i>	SQPI_PSRAM peripheral
<i>TZPCU_TZIAC_FLAG_QSPI_FLASH</i>	QSPI_FLASH peripheral

<i>QSPI_FLASH</i>	
<i>TZPCU_TZIAC_FLAG_</i> <i>SQPI_PSRAMREG</i>	SQPI PSRAMREG peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>QSPI_FLASHREG</i>	QSPI FLASHREG peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>WIFI_RF</i>	WIFI RF peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>I2S1_ADD</i>	I2S1_ADD peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>DCI</i>	DCI peripheral(DCI not support on GD32W515Tx series devices)
<i>TZPCU_TZIAC_FLAG_</i> <i>WIFI</i>	WIFI peripheral
Output parameter{out}	
-	-
Return value	
uint32_t	NO_ILLEGAL_ACCESS_PENDING or ILLEGAL_ACCESS_PENDING

Example:

```
/* get TIMER1 llegal access interrupt flag */
```

```
uint32_t tziac_flag = tzpcu_tziac_flag_get(TZPCU_TZIAC_FLAG_TIMER1);
```

tzpcu_tziac_flag_clear

The description of tzpcu_tziac_flag_clear is shown as below:

Table 3-1041. Function tzpcu_tziac_flag_clear

Function name	tzpcu_tziac_flag_clear
Function prototype	void tzpcu_tziac_flag_clear (uint32_t periph_flag);
Function descriptions	clear llegal access interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
periph_flag	peripheral interrupt flag
<i>TZPCU_TZIAC_FLAG_</i> <i>TIMER1</i>	TIMER1 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TIMER2</i>	TIMER2 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TIMER3</i>	TIMER3 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TIMER4</i>	TIMER4 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TIMER5</i>	TIMER5 peripheral

<i>TIMER5</i>	
<i>TZPCU_TZIAC_FLAG_WWDGT</i>	WWDGT peripheral
<i>TZPCU_TZIAC_FLAG_FWDGT</i>	FWDGT peripheral
<i>TZPCU_TZIAC_FLAG_SPI1</i>	SPI1 peripheral
<i>TZPCU_TZIAC_FLAG_USART1</i>	USART1 peripheral
<i>TZPCU_TZIAC_FLAG_USART2</i>	USART2 peripheral
<i>TZPCU_TZIAC_FLAG_I2C0</i>	I2C0 peripheral
<i>TZPCU_TZIAC_FLAG_I2C1</i>	I2C1 peripheral
<i>TZPCU_TZIAC_FLAG_USBFS</i>	USBFS peripheral
<i>TZPCU_TZIAC_FLAG_TIMER0</i>	TIMER0 peripheral
<i>TZPCU_TZIAC_FLAG_SPI0</i>	SPI0 peripheral
<i>TZPCU_TZIAC_FLAG_USART0</i>	USART0 peripheral
<i>TZPCU_TZIAC_FLAG_TIMER15</i>	TIMER15 peripheral
<i>TZPCU_TZIAC_FLAG_TIMER16</i>	TIMER16 peripheral
<i>TZPCU_TZIAC_FLAG_HPDF</i>	HPDF peripheral(HPDF not support on GD32W515Tx series devices)
<i>TZPCU_TZIAC_FLAG_CRC</i>	CRC peripheral
<i>TZPCU_TZIAC_FLAG_TSI</i>	TSI peripheral
<i>TZPCU_TZIAC_FLAG_ICACHE</i>	ICACHE peripheral
<i>TZPCU_TZIAC_FLAG_ADC</i>	ADC peripheral
<i>TZPCU_TZIAC_FLAG_CAU</i>	CAU peripheral
<i>TZPCU_TZIAC_FLAG_HAU</i>	HAU peripheral
<i>TZPCU_TZIAC_FLAG_TRNG</i>	TRNG peripheral

<i>TZPCU_TZIAC_FLAG_</i> <i>PKCAU</i>	PKCAU peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>SDIO</i>	SDIO peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>RTC</i>	RTC peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>PMU</i>	PMU peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>SYSCFG</i>	SYSCFG peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>DMA0</i>	DMA0 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>DMA1</i>	DMA1 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>RCU</i>	RCU peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>FLASH</i>	FLASH peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>FMC</i>	FLASH REG peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>EXTI</i>	EXTI peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TZSPC</i>	TZSPC peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TZIAC</i>	TZIAC peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>SRAM0</i>	SRAM0 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TZBMPC0_REG</i>	TZBMPC0 register
<i>TZPCU_TZIAC_FLAG_</i> <i>SRAM1</i>	SRAM1 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TZBMPC1_REG</i>	TZBMPC1 register
<i>TZPCU_TZIAC_FLAG_</i> <i>SRAM2</i>	SRAM2 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TZBMPC2_REG</i>	TZBMPC2 register
<i>TZPCU_TZIAC_FLAG_</i> <i>SRAM3</i>	SRAM3 peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>TZBMPC3_REG</i>	TZBMPC3 register
<i>TZPCU_TZIAC_FLAG_</i>	EFUSE peripheral

<i>EFUSE</i>	
<i>TZPCU_TZIAC_FLAG_</i> <i>SQPI_PSRAM</i>	SQPI_PSRAM peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>QSPI_FLASH</i>	QSPI_FLASH peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>SQPI_PSRAMREG</i>	SQPI PSRAMREG peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>QSPI_FLASHREG</i>	QSPI FLASHREG peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>WIFI_RF</i>	WIFI RF peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>I2S1_ADD</i>	I2S1_ADD peripheral
<i>TZPCU_TZIAC_FLAG_</i> <i>DCI</i>	DCI peripheral(DCI not support on GD32W515Tx series devices)
<i>TZPCU_TZIAC_FLAG_</i> <i>WIFI</i>	WIFI peripheral
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TIMER1 illegal access interrupt flag */
```

```
tzpcu_tziac_flag_clear(TZPCU_TZIAC_FLAG_TIMER1);
```

3.31. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter [3.31.1](#), the USART firmware functions are introduced in chapter [3.31.2](#).

3.31.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-1042. USART Registers

Registers	Descriptions
USART_CTL0	Control register 0
USART_CTL1	Control register 1
USART_CTL2	Control register 2
USART_BAUD	Baud rate register

Registers	Descriptions
USART_GP	Guard time and prescaler register
USART_RT	Receiver timeout register
USART_CMD	Command register
USART_STAT	Status register
USART_INTC	Status clear register
USART_RDATA	Receive data register
USART_TDATA	Transmit data register
USART_CHC	Coherence control register
USART_RFCS	Receive FIFO control and status register

3.31.2. Descriptions of Peripheral functions

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

Table 3-1043. USART firmware function

Function name	Function description
usart_deinit	reset USART
usart_baudrate_set	configure USART baud rate value
usart_parity_config	configure USART parity function
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_overrun_enable	enable the USART overrun function
usart_overrun_disable	disable the USART overrun function
usart_oversample_config	configure the USART oversample mode
usart_sample_bit_config	configure sample bit method
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
usart_address_config	configure address of the USART
usart_address_detection_mode_config	configure address detection mode
usart_mute_mode_enable	enable mute mode
usart_mute_mode_disable	disable mute mode

Function name	Function description
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_detection_length_config	LIN break detection length
usart_halfduplex_enable	enable half-duplex mode
usart_halfduplex_disable	disable half-duplex mode
usart_clock_enable	enable clock
usart_clock_disable	disable clock
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_mode_early_nack_enable	enable early NACK in smartcard mode
usart_smartcard_mode_early_nack_disable	disable early 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 or SmartCard 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_hardware_flow_coherence_config	configure hardware flow control coherence mode
usart_rs485_driver_enable	enable RS485 driver
usart_rs485_driver_disable	disable RS485 driver
usart_driver_asserttime_config	configure driver enable assertion time
usart_driver_deasserttime_config	configure driver enable de-assertion time
usart_depolarity_config	configure driver enable polarity mode
usart_dma_receive_config	configure USART DMA reception
usart_dma_transmit_config	configure USART DMA transmission
usart_reception_error_dma_disable	disable DMA on reception error
usart_reception_error_dma_enable	enable DMA on reception error
usart_wakeup_enable	enable USART to wakeup the MCU from deep-sleep mode
usart_wakeup_disable	disable USART to wakeup the MCU from deep-sleep mode

Function name	Function description
usart_wakeup_mode_config	configure the USART wakeup mode from deep-sleep mode
usart_receive_fifo_enable	enable receive FIFO
usart_receive_fifo_disable	disable receive FIFO
usart_receive_fifo_counter_number	read receive FIFO counter number
usart_command_enable	enable USART command
usart_flag_get	get flag in STAT/RFCs register
usart_flag_clear	clear USART status
usart_interrupt_enable	enable USART interrupt
usart_interrupt_disable	disable USART interrupt
usart_interrupt_flag_get	get USART interrupt and flag status
usart_interrupt_flag_clear	clear USART interrupt flag

Enum usart_flag_enum

Table 3-1044. Enum usart_flag_enum

Member name	Function description
USART_FLAG_REA	receive enable acknowledge flag
USART_FLAG_TEA	transmit enable acknowledge flag
USART_FLAG_WU	wakeup from Deep-sleep mode flag
USART_FLAG_RWU	receiver wakeup from mute mode
USART_FLAG_SB	send break flag
USART_FLAG_AM	ADDR match flag
USART_FLAG_BSY	busy flag
USART_FLAG_EB	end of block flag
USART_FLAG_RT	receiver timeout flag
USART_FLAG_CTS	CTS level
USART_FLAG_CTSF	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
USART_FLAG_NERR	noise error flag
USART_FLAG_FERR	frame error flag
USART_FLAG_PERR	parity error flag
USART_FLAG_EPERR	early parity error flag
USART_FLAG_RFFINT	receive FIFO full interrupt flag
USART_FLAG_RFF	receive FIFO full flag
USART_FLAG_RFE	receive FIFO empty flag

Enum usart_interrupt_flag_enum

Table 3-1045. Enum usart_interrupt_flag_enum

Member name	Function description
USART_INT_FLAG_EB	end of block interrupt and flag
USART_INT_FLAG_RT	receiver timeout interrupt and flag
USART_INT_FLAG_AM	address match interrupt and flag
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_ORE RR	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_WU	wakeup from deep-sleep mode interrupt and flag
USART_INT_FLAG_CTS	CTS interrupt and flag
USART_INT_FLAG_ERR_NERR	error interrupt and noise error flag
USART_INT_FLAG_ERR_ORER R	error interrupt and overrun error
USART_INT_FLAG_ERR_FERR	error interrupt and frame error flag
USART_INT_FLAG_RFF	receive FIFO full interrupt and flag

Enum usart_interrupt_enum

Table 3-1046. Enum usart_interrupt_enum

Member name	Function description
USART_INT_EB	end of block interrupt
USART_INT_RT	receiver timeout interrupt
USART_INT_AM	address match interrupt
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_WU	wakeup from deep-sleep mode interrupt
USART_INT_CTS	CTS interrupt
USART_INT_ERR	error interrupt
USART_INT_RFF	receive FIFO full interrupt

Enum usart_invert_enum

Table 3-1047. Enum 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_SWAP_ENABLE	swap TX/RX pins
USART_SWAP_DISABLE	not swap TX/RX pins

usart_deinit

The description of usart_deinit is shown as below:

Table 3-1048. Function usart_deinit

Function name	usart_deinit
Function prototype	void usart_deinit(uint32_t usart_periph);
Function descriptions	reset USART
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
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 3-1049. 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_clock_freq_get
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
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 3-1050. 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 peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
paritycfg	configure USART parity
<i>USART_PM_NONE</i>	no parity
<i>USART_PM_ODD</i>	odd parity
<i>USART_PM_EVEN</i>	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 3-1051. 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 peripheral
USARTx	x=0,1,2
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 3-1052. 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 peripheral
USARTx	x=0,1,2
Input parameter{in}	
stblen	USART stop bit
USART_STB_1BIT	1 bit
USART_STB_0_5BIT	0.5 bit
USART_STB_2BIT	2 bits

<i>USART_STB_1_5BIT</i>	1.5 bits
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 3-1053. 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 peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 */
```

```
usart_enable(USART0);
```

usart_disable

The description of usart_disable is shown as below:

Table 3-1054. 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 peripheral

<i>USARTx</i>	<i>x=0,1,2</i>
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 3-1055. 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 peripheral
<i>USARTx</i>	<i>x=0,1,2</i>
Input parameter{in}	
txconfig	enable or disable USART transmitter
<i>USART_TRANSMIT_ENABLE</i>	enable USART transmission
<i>USART_TRANSMIT_DISABLE</i>	disable 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 3-1056. 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 peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
rxconfig	enable or disable USART receiver
<i>USART_RECEIVE_ENABLE</i>	enable USART reception
<i>USART_RECEIVE_DISABLE</i>	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 3-1057. 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
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
msbf	LSB/MSB
<i>USART_MSBF_LSB</i>	LSB first
<i>USART_MSBF_MSB</i>	MSB first
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 data is transmitted/received with the LSB first */
```

```
usart_data_first_config(USART0, USART_MSBF_LSB);
```

usart_invert_config

The description of usart_invert_config is shown as below:

Table 3-1058. 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	USART inverted configure
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
invertpara	refer to Table 3-1047. Enum usart_invert_enum
USART_DINV_ENAB LE	data bit level inversion
USART_DINV_DISAB LE	data bit level not inversion
USART_TXPIN_ENAB LE	TX pin level inversion
USART_TXPIN_DISAB LE	TX pin level not inversion
USART_RXPIN_ENAB LE	RX pin level inversion
USART_RXPIN_DISAB LE	RX pin level not inversion
USART_SWAP_ENAB LE	swap TX/RX pins
USART_SWAP_DISAB LE	not swap TX/RX pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 data bit level inversion */
```

usart_invert_config (USART0, USART_DINV_ENABLE);

usart_oversize_enable

The description of usart_oversize_enable is shown as below:

Table 3-1059. Function usart_oversize_enable

Function name	usart_oversize_enable
Function prototype	void usart_oversize_enable(uint32_t usart_periph);
Function descriptions	enable the USART oversize function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the USART0 oversize function */
```

```
usart_oversize_enable(USART0);
```

usart_oversize_disable

The description of usart_oversize_disable is shown as below:

Table 3-1060. Function usart_oversize_disable

Function name	usart_oversize_disable
Function prototype	void usart_oversize_disable(uint32_t usart_periph);
Function descriptions	disable the USART oversize function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the USART0 oversize function */
```

```
usart_oversample_disable(USART0);
```

usart_oversample_config

The description of usart_oversample_config is shown as below:

Table 3-1061. Function usart_oversample_config

Function name	usart_oversample_config
Function prototype	void usart_oversample_config(uint32_t usart_periph, uint32_t oversamp);
Function descriptions	configure the USART oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
oversamp	oversample value
<i>USART_OVSMOD_8</i>	8 bits
<i>USART_OVSMOD_16</i>	16 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 oversample mode */
```

```
usart_oversample_config(USART0, USART_OVSMOD_8);
```

usart_sample_bit_config

The description of usart_sample_bit_config is shown as below:

Table 3-1062. Function usart_sample_bit_config

Function name	usart_sample_bit_config
Function prototype	void usart_sample_bit_config(uint32_t usart_periph, uint32_t osb);
Function descriptions	configure the sample bit method
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
osb	sample bit
<i>USART_OSB_1bit</i>	1 bits

<i>USART_OSB_3bit</i>	3 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 sample bit */
```

```
usart_sample_bit_config(USART0, USART_OSB_1bit);
```

usart_receiver_timeout_enable

The description of usart_receiver_timeout_enable is shown as below:

Table 3-1063. 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
<i>USARTx</i>	x=0,2
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 3-1064. 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=0,2
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 3-1065. 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 timeout);
Function descriptions	configure receiver timeout threshold
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,2
Input parameter{in}	
rtimeout	receiver timeout threshold (0x00000000 - 0x00FFFFFF)
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 3-1066. Function usart_data_transmit

Function name	usart_data_transmit
Function prototype	void usart_data_transmit(uint32_t usart_periph, uint32_t data);
Function descriptions	USART transmit data function
Precondition	-

The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
data	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_receive is shown as below:

Table 3-1067. 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 peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
uint32_t	data of received

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 3-1068. Function `usart_address_config`

Function name	<code>usart_address_config</code>
Function prototype	<code>void usart_address_config(uint32_t usart_periph, uint8_t addr);</code>
Function descriptions	address of the USART terminal
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	USARTx peripheral
<code>USARTx</code>	x=0,1,2
Input parameter{in}	
<code>addr</code>	address of USART terminal(0x00-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address of USART0 */
```

```
usart_address_config(USART0, 0x00);
```

`usart_address_detection_mode_config`

The description of `usart_address_detection_mode_config` is shown as below:

Table 3-1069. Function `usart_address_detection_mode_config`

Function name	<code>usart_address_detection_mode_config</code>
Function prototype	<code>void usart_address_detection_mode_config(uint32_t usart_periph, uint32_t addmod);</code>
Function descriptions	configure address detection mode
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	USARTx peripheral
<code>USARTx</code>	x=0,1,2
Input parameter{in}	
<code>addmod</code>	address detection mode
<code>USART_ADDDM_4BIT</code>	4 bits
<code>USART_ADDDM_FULLBIT</code>	full bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address detection mode */

usart_address_config(USART0, USART_ADDDM_4BIT);
```

usart_mute_mode_enable

The description of usart_mute_mode_enable is shown as below:

Table 3-1070. 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 peripheral
<i>USARTx</i>	x=0,1,2
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 3-1071. 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 peripheral
<i>USARTx</i>	x=0,1,2
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 3-1072. 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 peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
wmethod	two methods be used to enter or exit the mute mode
<i>USART_WM_IDLE</i>	idle line
<i>USART_WM_ADDR</i>	address mark
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 3-1073. 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 peripheral

USARTx	x=0,2
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 3-1074. 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 peripheral
USARTx	x=0,2
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 3-1075. 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 lflen);
Function descriptions	configure LIN break frame length
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,2
Input parameter{in}	
lblen	two methods be used to enter or exit the mute mode
<i>USART_LBLEN_10B</i>	break frame length is 10 bits
<i>USART_LBLEN_11B</i>	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_halfduplex_enable

The description of usart_halfduplex_enable is shown as below:

Table 3-1076. 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 peripheral
<i>USARTx</i>	x=0,1,2
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 3-1077. 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 peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 half duplex mode*/
usart_halfduplex_disable(USART0);
```

usart_clock_enable

The description of usart_clock_enable is shown as below:

Table 3-1078. Function usart_clock_enable

Function name	usart_clock_enable
Function prototype	void usart_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
<i>USARTx</i>	x=0,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 CK pin in synchronous mode */
usart_clock_enable(USART0);
```

usart_clock_disable

The description of usart_clock_disable is shown as below:

Table 3-1079. Function usart_clock_disable

Function name	usart_clock_disable
Function prototype	void usart_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
<i>USARTx</i>	x=0,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 CK pin in synchronous mode */
```

```
usart_clock_disable(USART0);
```

usart_synchronous_clock_config

The description of usart_synchronous_clock_config is shown as below:

Table 3-1080. 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
<i>USARTx</i>	x=0,2
Input parameter{in}	
clen	CK length
<i>USART_CLEN_NONE</i>	clock pulse of the last data bit (MSB) is not output to the CK pin
<i>USART_CLEN_EN</i>	clock pulse of the last data bit (MSB) is output to the CK pin
Input parameter{in}	
cph	clock phase
<i>USART_CPH_1CK</i>	first clock transition is the first data capture edge
<i>USART_CPH_2CK</i>	second clock transition is the first data capture edge
Input parameter{in}	
cpl	clock polarity
<i>USART_CPL_LOW</i>	steady low value on CK pin
<i>USART_CPL_HIGH</i>	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 3-1081. Function usart_guard_time_config

Function name	usart_guard_time_config
Function prototype	void usart_guard_time_config(uint32_t usart_periph, uint32_t gaut);
Function descriptions	configure guard time value in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,2
Input parameter{in}	
gaut	guard time value (0x00 - 0xFF)
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 3-1082. 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

<i>USARTx</i>	<i>x=0,2</i>
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 3-1083. 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
<i>USARTx</i>	<i>x=0,2</i>
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 3-1084. 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=0,2
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 3-1085. 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=0,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 NACK in smartcard mode */
usart_smartcard_mode_nack_disable(USART0);
```

usart_smartcard_mode_early_nack_enable

The description of usart_smartcard_mode_early_nack_enable is shown as below:

Table 3-1086. Function usart_smartcard_mode_early_nack_enable

Function name	usart_smartcard_mode_early_nack_enable
Function prototype	void usart_smartcard_mode_early_nack_enable(uint32_t usart_periph);
Function descriptions	enable early NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral

<i>USARTx</i>	<i>x=0,2</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 early NACK in smartcard mode */
usart_smartcard_mode_early_nack_enable(USART0);
```

usart_smartcard_mode_early_nack_disable

The description of usart_smartcard_mode_early_nack_disable is shown as below:

Table 3-1087. Function usart_smartcard_mode_early_nack_disable

Function name	usart_smartcard_mode_early_nack_disable
Function prototype	void usart_smartcard_mode_early_nack_disable(uint32_t usart_periph);
Function descriptions	disable early NACK in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	<i>x=0,2</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 early NACK in smartcard mode */
usart_smartcard_mode_early_nack_disable(USART0);
```

usart_smartcard_autoretry_config

The description of usart_smartcard_autoretry_config is shown as below:

Table 3-1088. Function usart_smartcard_autoretry_config

Function name	usart_smartcard_autoretry_config
Function prototype	void usart_smartcard_autoretry_config(uint32_t usart_periph, uint32_t scrtnum);
Function descriptions	configure smartcard auto-retry number
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,2
Input parameter{in}	
scrtnum	smartcard auto-retry number (0x00 - 0x07)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure smartcard auto-retry number */
usart_smartcard_autoretry_config (USART0, 0x07);
```

usart_block_length_config

The description of usart_block_length_config is shown as below:

Table 3-1089. Function usart_block_length_config

Function name	usart_block_length_config
Function prototype	void usart_block_length_config(uint32_t usart_periph, uint32_t bl);
Function descriptions	configure block length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,2
Input parameter{in}	
bl	block length (0x00 - 0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure block length in Smartcard T=1 reception */
usart_block_length_config(USART0, 0x000000FF);
```

usart_irda_mode_enable

The description of usart_irda_mode_enable is shown as below:

Table 3-1090. 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 peripheral
<i>USARTx</i>	x=0,2
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 3-1091. 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 peripheral
<i>USARTx</i>	x=0,2
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 3-1092. 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 peripheral
<i>USARTx</i>	x=0,2
Input parameter{in}	
psc	0x00000000 - 0x000000FF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the USART0 peripheral clock prescaler */
```

```
usart_prescaler_config(USART0, 0x00000000);
```

usart_irda_lowpower_config

The description of usart_irda_lowpower_config is shown as below:

Table 3-1093. 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 peripheral
<i>USARTx</i>	x=0,2
Input parameter{in}	
irlp	IrDA low-power or normal
<i>USART_IRLP_LOW</i>	low-power
<i>USART_IRLP_NORMAL</i>	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 3-1094. 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
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
rtsconfig	enable or disable RTS
<i>USART_RTS_ENABLE</i>	enable RTS
<i>USART_RTS_DISABLE</i>	disable RTS
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 hardware flow control RTS */
usart_hardware_flow_rts_config(USART0, USART_RTS_ENABLE);
```

usart_hardware_flow_cts_config

The description of usart_hardware_flow_cts_config is shown as below:

Table 3-1095. 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
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
ctsconfig	enable or disable CTS

<i>USART_CTS_ENABLE</i>	enable CTS
<i>USART_CTS_DISABLE</i>	disable CTS
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 hardware flow control CTS */
```

```
usart_hardware_flow_cts_config(USART0, USART_CTS_ENABLE);
```

usart_hardware_flow_coherence_config

The description of usart_hardware_flow_coherence_config is shown as below:

Table 3-1096. 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 peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
hcm	hardware flow control coherence mode
<i>USART_HCM_NONE</i>	nRTS signal equals to the rbne status register
<i>USART_HCM_EN</i>	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_HCM_NONE);
```

usart_rs485_driver_enable

The description of usart_rs485_driver_enable is shown as below:

Table 3-1097. Function usart_rs45_driver_enable

Function name	usart_rs485_driver_enable
Function prototype	void usart_rs485_driver_enable(uint32_t usart_periph);
Function descriptions	enable RS485 driver
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 RS485 driver */
usart_rs485_driver_enable(USART0);
```

usart_rs485_driver_disable

The description of usart_rs485_driver_disable is shown as below:

Table 3-1098. Function usart_rs45_driver_disable

Function name	usart_rs485_driver_disable
Function prototype	void usart_rs485_driver_disable(uint32_t usart_periph);
Function descriptions	disable RS485 driver
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 RS485 driver */
usart_rs485_driver_disable(USART0);
```

usart_driver_assertime_config

The description of usart_driver_assertime_config is shown as below:

Table 3-1099. Function `usart_driver_assertime_config`

Function name	<code>usart_driver_assertime_config</code>
Function prototype	<code>void usart_driver_assertime_config(uint32_t usart_periph, uint32_t deatime);</code>
Function descriptions	configure driver enable assertion time
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	USARTx peripheral
<code>USARTx</code>	x=0,1,2
Input parameter{in}	
<code>deatime</code>	driver enable assertion time(0x00-0x1F)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set USART0 driver assertime */
usart_driver_assertime_config(USART0, 0x1F);
```

`usart_driver_deassertime_config`

The description of `usart_driver_deassertime_config` is shown as below:

Table 3-1100. Function `usart_driver_deassertime_config`

Function name	<code>usart_driver_deassertime_config</code>
Function prototype	<code>void usart_driver_deassertime_config(uint32_t usart_periph, uint32_t dedtime);</code>
Function descriptions	configure driver enable de-assertion time
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	USARTx peripheral
<code>USARTx</code>	x=0,1,2
Input parameter{in}	
<code>deatime</code>	driver enable deassertime (0x00-0x1F)
Output parameter{out}	
-	-
Return value	
-	-

Example:


```
/* set USART0 driver deasserttime */
```

```
usart_driver_deassertime_config(USART0, 0x1F);
```

usart_depolarity_config

The description of usart_depolarity_config is shown as below:

Table 3-1101. Function usart_depolarity_config

Function name	usart_depolarity_config
Function prototype	void usart_depolarity_config(uint32_t usart_periph, uint32_t dep);
Function descriptions	configure driver enable polarity mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
<i>dep</i>	DE signal
<i>USART_DEP_HIGH</i>	DE signal is active high
<i>USART_DEP_LOW</i>	DE signal is active low
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure driver enable polarity mode */
```

```
usart_driver_depolarity_config(USART0, USART_DEP_HIGH);
```

usart_dma_receive_config

The description of usart_dma_receive_config is shown as below:

Table 3-1102. Function usart_dma_receive_config

Function name	usart_dma_receive_config
Function prototype	void usart_dma_receive_config(uint32_t usart_periph, uint8_t dmacmd);
Function descriptions	configure USART DMA reception
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
dmacmd	USART DMA mode

<i>USART_RECEIVE_DMA_ENABLE</i>	enable USART DMA for reception
<i>USART_RECEIVE_DMA_DISABLE</i>	disable USART DMA for reception
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART DMA 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 3-1103. Function usart_dma_transmit_config

Function name	usart_dma_transmit_config
Function prototype	void usart_dma_transmit_config(uint32_t usart_periph, uint8_t dmacmd);
Function descriptions	configure USART DMA transmission
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
dmacmd	USART DMA mode
<i>USART_TRANSMIT_DMA_ENABLE</i>	enable USART DMA for transmission
<i>USART_TRANSMIT_DMA_DISABLE</i>	disable USART DMA for transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART DMA transmission */
usart_dma_transmit_config(USART0, USART_TRANSMIT_DMA_ENABLE);
```

usart_reception_error_dma_disable

The description of usart_reception_error_dma_disable is shown as below:

Table 3-1104. Function usart_reception_error_dma_disable

Function name	usart_reception_error_dma_disable
Function prototype	void usart_reception_error_dma_disable(uint32_t usart_periph);
Function descriptions	disable DMA on reception error
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA on reception error */
usart_reception_error_dma_disable (USART0);
```

usart_reception_error_dma_enable

The description of usart_reception_error_dma_enable is shown as below:

Table 3-1105. Function usart_reception_error_dma_enable

Function name	usart_reception_error_dma_enable
Function prototype	void usart_reception_error_dma_enable(uint32_t usart_periph);
Function descriptions	enable DMA on reception error
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA on reception error */
usart_reception_error_dma_enable (USART0);
```

usart_wakeup_enable

The description of usart_wakeup_enable is shown as below:

Table 3-1106. Function usart_wakeup_enable

Function name	usart_wakeup_enable
Function prototype	void usart_wakeup_enable(uint32_t usart_periph);
Function descriptions	enable USART to wakeup the mcu from deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x=0,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 wake up enable */
usart_wakeup_enable(USART0);
```

usart_wakeup_disable

The description of usart_wakeup_disable is shown as below:

Table 3-1107. Function usart_wakeup_disable

Function name	usart_wakeup_disable
Function prototype	void usart_wakeup_disable(uint32_t usart_periph);
Function descriptions	disable USART to wakeup the mcu from deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x=0,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 wake up disable */
usart_wakeup_disable(USART0);
```

usart_wakeup_mode_config

The description of usart_wakeup_mode_config is shown as below:

Table 3-1108. Function usart_wakeup_mode_config

Function name	usart_wakeup_mode_config
Function prototype	void usart_wakeup_mode_config(uint32_t usart_periph, uint32_t wum);
Function descriptions	configure the USART wakeup mode from deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,2
Input parameter{in}	
wum	wakeup mode
USART_WUM_ADDR	WUF active on address match
USART_WUM_START B	WUF active on start bit
USART_WUM_RBNE	WUF active on RBNE
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART wake up mode */
```

```
usart_wakeup_mode_config(USART, USART_WUM_ADDR);
```

usart_receive_fifo_enable

The description of usart_receive_fifo_enable is shown as below:

Table 3-1109. Function usart_receive_fifo_enable

Function name	usart_receive_fifo_enable
Function prototype	void usart_receive_fifo_enable(uint32_t usart_periph);
Function descriptions	enable receive FIFO
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,2
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable receive FIFO */
```

```
usart_receive_fifo_enable(USART0);
```

usart_receive_fifo_disable

The description of usart_receive_fifo_disable is shown as below:

Table 3-1110. Function usart_receive_fifo_disable

Function name	usart_receive_fifo_disable
Function prototype	void usart_receive_fifo_disable(uint32_t usart_periph);
Function descriptions	disable receive FIFO
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable receive FIFO */
```

```
usart_receive_fifo_disable(USART0);
```

usart_receive_fifo_counter_number

The description of usart_receive_fifo_counter_number is shown as below:

Table 3-1111. Function usart_receive_fifo_counter_number

Function name	usart_receive_fifo_counter_number
Function prototype	uint8_t usart_receive_fifo_counter_number(uint32_t usart_periph);
Function descriptions	read receive FIFO counter number
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	

uint8_t	receive FIFO counter number
----------------	-----------------------------

Example:

```
/* read receive FIFO counter number */

uint8_t temp;

temp = usart_receive_fifo_counter_number(USART0);
```

usart_command_enable

The description of usart_command_enable is shown as below:

Table 3-1112. Function usart_command_enable

Function name	usart_command_enable
Function prototype	void usart_command_enable(uint32_t usart_periph, uint32_t cmdtype);
Function descriptions	enable USART command
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Input parameter{in}	
cmdtype	command type
USART_CMD_SBKCMD	send break command
USART_CMD_MMCMMD	mute mode command
USART_CMD_RXFCM	receive data flush command
USART_CMD_TXFCM	transmit data flush request
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 command */

usart_command_enable(USART0, USART_CMD_SBKCMD);
```

usart_flag_get

The description of usart_flag_get is shown as below:

Table 3-1113. 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 STAT/CHC/RFCR register
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
flag	USART flags, refer to Table 3-1044. Enum usart_flag_enum
USART_FLAG_PERR	parity error flag
USART_FLAG_FERR	frame error flag
USART_FLAG_NERR	noise error flag
USART_FLAG_ORER R	overrun error
USART_FLAG_IDLE	idle line detected flag
USART_FLAG_RBNE	read data buffer not empty
USART_FLAG_TC	transmission completed
USART_FLAG_TBE	transmit data register empty
USART_FLAG_LBD	LIN break detected flag
USART_FLAG_CTSF	CTS change flag
USART_FLAG_CTS	CTS level
USART_FLAG_RT	receiver timeout flag
USART_FLAG_EB	end of block flag
USART_FLAG_BSY	busy flag
USART_FLAG_AM	address match flag
USART_FLAG_SB	send break flag
USART_FLAG_RWU	receiver wakeup from mute mode
USART_FLAG_WU	wakeup from deep-sleep mode flag
USART_FLAG_TEA	transmit enable acknowledge flag
USART_FLAG_REA	receive enable acknowledge flag
USART_FLAG_EPERR	early parity error flag
USART_FLAG_RFE	receive FIFO empty flag
USART_FLAG_RFF	receive FIFO full flag
USART_FLAG_RFFINT	receive FIFO full interrupt flag
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 3-1114. 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 USART status
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
flag	USART flags, refer to Table 3-1044. Enum usart_flag_enum
<i>USART_FLAG_PERR</i>	parity error flag
<i>USART_FLAG_FERR</i>	frame error flag
<i>USART_FLAG_NERR</i>	noise detected flag
<i>USART_FLAG_ORER R</i>	overrun error flag
<i>USART_FLAG_IDLE</i>	idle line detected flag
<i>USART_FLAG_TC</i>	transmission complete flag
<i>USART_FLAG_LBD</i>	LIN break detected flag
<i>USART_FLAG_CTSF</i>	CTS change flag
<i>USART_FLAG_RT</i>	receiver timeout flag
<i>USART_FLAG_EB</i>	end of block flag
<i>USART_FLAG_AM</i>	address match flag
<i>USART_FLAG_WU</i>	wakeup from deep-sleep mode flag
<i>USART_FLAG_EPERR</i>	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_enable is shown as below:

Table 3-1115. 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 peripheral
USARTx	x=0,1,2
Input parameter{in}	
interrupt	USART interrupt, refer to Table 3-1046. Enum usart_interrupt_enum
USART_INT_IDLE	idle interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt enable interrupt
USART_INT_TC	transmission complete interrupt
USART_INT_TBE	transmit data register empty interrupt
USART_INT_PERR	parity error interrupt
USART_INT_AM	address match interrupt
USART_INT_RT	receiver timeout interrupt
USART_INT_EB	end of block interrupt
USART_INT_LBD	LIN break detection interrupt
USART_INT_ERR	error interrupt enable in multibuffer communication
USART_INT_CTS	CTS interrupt
USART_INT_WU	wakeup from deep-sleep mode interrupt
USART_INT_RFF	receive FIFO full interrupt enable
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 3-1116. 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 peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
intterrupt	USART interrupt flag, refer to Table 3-1046. Enum usart_interrupt_enum
<i>USART_INT_IDLE</i>	idle interrupt
<i>USART_INT_RBNE</i>	read data buffer not empty interrupt and overrun error interrupt enable interrupt
<i>USART_INT_TC</i>	transmission complete interrupt
<i>USART_INT_TBE</i>	transmit data register empty interrupt
<i>USART_INT_PERR</i>	parity error interrupt
<i>USART_INT_AM</i>	address match interrupt
<i>USART_INT_RT</i>	receiver timeout interrupt
<i>USART_INT_EB</i>	end of block interrupt
<i>USART_INT_LBD</i>	LIN break detection interrupt
<i>USART_INT_ERR</i>	error interrupt enable in multibuffer communication
<i>USART_INT_CTS</i>	CTS interrupt
<i>USART_INT_WU</i>	wakeup from deep-sleep mode interrupt
<i>USART_INT_RFF</i>	receive FIFO full interrupt enable
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 3-1117. 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
<i>USARTx</i>	x=0,1,2
<i>UARTx</i>	x=3,4
Input parameter{in}	
int_flag	USART interrupt flag, refer to Table 3-1045. Enum usart_interrupt_flag_enum
<i>USART_INT_FLAG_EB</i>	end of block interrupt and flag
<i>USART_INT_FLAG_RT</i>	receiver timeout interrupt and flag
<i>USART_INT_FLAG_A M</i>	address match interrupt and flag
<i>USART_INT_FLAG_PE RR</i>	parity error interrupt and flag
<i>USART_INT_FLAG_TB E</i>	transmitter buffer empty interrupt and flag
<i>USART_INT_FLAG_TC</i>	transmission complete interrupt and flag
<i>USART_INT_FLAG_RB NE</i>	read data buffer not empty interrupt and flag
<i>USART_INT_FLAG_RB NE_ORERR</i>	read data buffer not empty interrupt and overrun error flag
<i>USART_INT_FLAG_ID LE</i>	IDLE line detected interrupt and flag
<i>USART_INT_FLAG_LB D</i>	LIN break detected interrupt and flag
<i>USART_INT_FLAG_W U</i>	wakeup from deep-sleep mode interrupt and flag
<i>USART_INT_FLAG_CT S</i>	CTS interrupt and flag
<i>USART_INT_FLAG_ER R_NERR</i>	error interrupt and noise error flag
<i>USART_INT_FLAG_ER R_ORERR</i>	error interrupt and overrun error
<i>USART_INT_FLAG_ER R_FERR</i>	error interrupt and frame error flag
<i>USART_INT_FLAG_RF F</i>	receive FIFO full interrupt and 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 3-1118. 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 peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
int_flag	USART interrupt flag, refer to Table 3-1045. Enum usart_interrupt_flag_enum
<i>USART_INT_FLAG_PERR</i>	parity error flag
<i>USART_INT_FLAG_ERFERR</i>	frame error flag
<i>USART_INT_FLAG_ER_NERR</i>	noise detected flag
<i>USART_INT_FLAG_RBNE_ORERR</i>	read data buffer not empty interrupt and overrun error flag
<i>USART_INT_FLAG_ER_ORERR</i>	error interrupt and overrun error
<i>USART_INT_FLAG_IDLE</i>	idle line detected flag
<i>USART_INT_FLAG_TC</i>	transmission complete flag
<i>USART_INT_FLAG_LBD</i>	LIN break detected flag
<i>USART_INT_FLAG_CTS</i>	CTS change flag
<i>USART_INT_FLAG_RTS</i>	receiver timeout flag
<i>USART_INT_FLAG_EB</i>	end of block flag

USART_INT_FLAG_A M	address match flag
USART_INT_FLAG_W U	wakeup from deep-sleep mode flag
USART_INT_FLAG_RF F	receive FIFO full interrupt and flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the USART0 interrupt enable bit status */
```

```
usart_interrupt_flag_clear(USART0, USART_INT_FLAG_RBNE);
```

3.32. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. The WWDGT registers are listed in chapter [3.32.1](#), the WWDGT firmware functions are introduced in chapter [3.32.2](#).

3.32.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

Table 3-1119. WWDGT Registers

Registers	Descriptions
WWDGT_CTL	Control register
WWDGT_CFG	Configuration register
WWDGT_STAT	Status register

3.32.2. Descriptions of Peripheral functions

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

Table 3-1120. WWDGT firmware function

Function name	Function description
wwdgt_deinit	reset the WWDGT configuration
wwdgt_enable	start the WWDGT counter
wwdgt_counter_update	configure the WWDGT counter value
wwdgt_config	configure counter value, window value, and prescaler divider value
wwdgt_interrupt_enable	enable early wakeup interrupt of WWDGT

Function name	Function description
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 3-1121. Function wwdgt_deinit

Function name	wwdgt_deinit
Function prototype	void wwdgt_deinit(void);
Function descriptions	reset the WWDGT configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the WWDGT configuration */
```

```
wwdgt_deinit ( );
```

wwdgt_enable

The description of wwdgt_enable is shown as below:

Table 3-1122. Function wwdgt_enable

Function name	wwdgt_enable
Function prototype	void wwdgt_enable(void);
Function descriptions	start theWWDGT 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 3-1123. Function wwdgt_counter_update

Function name	wwdgt_counter_update
Function prototype	void wwdgt_counter_update(uint16_t counter_value);
Function descriptions	configure the WWDGT 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(0x7F);
```

wwdgt_config

The description of wwdgt_config is shown as below:

Table 3-1124. 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_D IV1	the time base of window watchdog counter = (PCLK1/4096)/1
WWDGT_CFG_PSC_D IV2	the time base of window watchdog counter = (PCLK1/4096)/2
WWDGT_CFG_PSC_D	the time base of window watchdog counter = (PCLK1/4096)/4

IV4	
WWDGT_CFG_PSC_D IV8	the time base of window watchdog counter = (PCLK1/4096)/8
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* confiure WWDGT counter value to 0x7F, window value to 0x50, prescaler divider value to 8 */
```

```
wwdgt_config(0x7F, 0x50, WWDGT_CFG_PSC_DIV8);
```

wwdgt_interrupt_enable

The description of wwdgt_interrupt_enable is shown as below:

Table 3-1125. 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 3-1126. 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 ( );
```

wwdgt_flag_clear

The description of wwdgt_flag_clear is shown as below:

Table 3-1127. 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

Revision No.	Description	Date
1.0	Initial Release	Nov.23, 2021
1.1	<p>1. Change function efuse_mcu_init_data_write / efuse_aes_key_write / efuse_rotpk_key_write / efuse_dp_write / efuse_iak_write / efuse_user_data_write in <u>3.8.2.</u></p> <p>2.Change function qspi_enable to spi_quad_enable / qspi_disable to spi_quad_disable / qspi_write_enable to spi_quad_write_enable / qspi_read_enable to spi_quad_read_enable / qspi_io23_output_enable to spi_quad_io23_output_enable / qspi_io23_output_disable to spi_quad_io23_output_disable in <u>3.24.2.</u></p> <p>3.Change function quad_spi_enable to qspi_enable / quad_spi_disable to qspi_disable in <u>3.20.2.</u></p> <p>4. Modify the function order in <u>3.21.2.</u></p>	Jul.5, 2022
1.2	<p>1.Add function spi_i2s_format_error_clear in <u>3.24.2</u></p> <p>2.Change function usart_dma_enable / usart_dma_disable to usart_dma_receive_config /usart_dma_transmit_config in <u>3.31.2</u></p> <p>3.Add enmu rcu_unit_enum, modify function interface rcu_xxx_poweron / rcu_xxx_powerdown to rcu_control_unit_powerup /rcu_control_unit_powerdown, change function rcu_plldigfsys_div_config to rcu_plldig_div_sys_config, change function rcu_pllfi2s_clock_div_config to rcu_pll_div_i2s_config, delet function rcu_plldig_enable/disable in <u>3.22.2</u></p>	Dec.25, 2022

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 under the intellectual property laws and treaties of the People's Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. 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.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do 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 Products. Customers shall indemnify and hold the Company 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 Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.