

GigaDevice Semiconductor Inc.

GD32A513

Arm[®] Cortex[®]-M33 32-bit MCU

**Firmware Library
User Guide**

Revision 1.0

(Aug. 2023)

Table of Contents

Table of Contents	1
List of Figures	4
List of Tables	5
1. Introduction	25
1.1. Rules of User Manual and Firmware Library	25
1.1.1. Peripherals.....	25
1.1.2. Naming rules.....	26
2. Firmware Library Overview	27
2.1. File Structure of Firmware Library	27
2.1.1. Examples Folder	28
2.1.2. Firmware Folder	28
2.1.3. Template Folder	28
2.1.4. Utilities Folder	30
2.2. File descriptions of Firmware Library	31
3. Firmware Library of Standard Peripherals	32
3.1. Overview of Firmware Library of Standard Peripherals.....	32
3.2. ADC	32
3.2.1. Descriptions of Peripheral registers.....	32
3.2.2. Descriptions of Peripheral functions	33
3.3. BKP.....	62
3.3.1. Descriptions of Peripheral registers.....	62
3.3.2. Descriptions of Peripheral functions	62
3.4. CAN	74
3.4.1. Descriptions of Peripheral registers.....	75
3.4.2. Descriptions of Peripheral functions	76
3.5. CMP	120
3.5.1. Descriptions of Peripheral registers.....	120
3.5.2. Descriptions of Peripheral functions	120
3.6. CRC	128
3.6.1. Descriptions of Peripheral registers.....	129
3.6.2. Descriptions of Peripheral functions	129
3.7. DBG	137
3.7.1. Descriptions of Peripheral registers.....	137

3.7.2.	Descriptions of Peripheral functions	137
3.8.	DAC	141
3.8.1.	Descriptions of Peripheral registers	141
3.8.2.	Descriptions of Peripheral functions	142
3.9.	DMA / DMAMUX	155
3.9.1.	Descriptions of Peripheral registers	155
3.9.2.	Descriptions of Peripheral functions	156
3.10.	EXTI.....	202
3.10.1.	Descriptions of Peripheral registers	202
3.10.2.	Descriptions of Peripheral functions	202
3.11.	FMC	210
3.11.1.	Descriptions of Peripheral registers	210
3.11.2.	Descriptions of Peripheral functions	210
3.12.	FWDGT.....	251
3.12.1.	Descriptions of Peripheral registers	252
3.12.2.	Descriptions of Peripheral functions	252
3.13.	GPIO.....	257
3.13.1.	Descriptions of Peripheral registers	257
3.13.2.	Descriptions of Peripheral functions	258
3.14.	I2C	268
3.14.1.	Descriptions of Peripheral registers	268
3.14.2.	Descriptions of Peripheral functions	269
3.15.	MFCOM	307
3.15.1.	Descriptions of Peripheral registers	307
3.15.2.	Descriptions of Peripheral functions	307
3.16.	MISC.....	331
3.16.1.	Descriptions of Peripheral registers	331
3.16.2.	Descriptions of Peripheral functions	332
3.17.	PMU.....	339
3.17.1.	Descriptions of Peripheral registers	339
3.17.2.	Descriptions of Peripheral functions	339
3.18.	RCU	350
3.18.1.	Descriptions of Peripheral registers	351
3.18.2.	Descriptions of Peripheral functions	351
3.19.	RTC	386
3.19.1.	Descriptions of Peripheral registers	386
3.19.2.	Descriptions of Peripheral functions	386
3.20.	SPI.....	396
3.20.1.	Descriptions of Peripheral registers	396

3.20.2.	Descriptions of Peripheral functions	396
3.21.	SYSCFG	422
3.21.1.	Descriptions of Peripheral registers	422
3.21.2.	Descriptions of Peripheral functions	423
3.22.	TIMER	434
3.22.1.	Descriptions of Peripheral registers	434
3.22.2.	Descriptions of Peripheral functions	435
3.23.	TRIGSEL	511
3.23.1.	Descriptions of Peripheral registers	511
3.23.2.	Descriptions of Peripheral functions	512
3.24.	USART	517
3.24.1.	Descriptions of Peripheral registers	517
3.24.2.	Descriptions of Peripheral functions	518
3.25.	WWDGT	563
3.25.1.	Descriptions of Peripheral registers	563
3.25.2.	Descriptions of Peripheral functions	563
4.	Revision history	569

List of Figures

Figure 2-1. File structure of firmware library of GD32A513	27
Figure 2-2. Select peripheral example files.....	29
Figure 2-3. Copy the peripheral example files	29
Figure 2-4. Open the project file.....	29
Figure 2-5. Configure project files	30
Figure 2-6. Compile-debug-download	30

List of Tables

Table 1-1. Peripherals	25
Table 2-1. Function descriptions of Firmware Library	31
Table 3-1. Peripheral function format of Firmware Library	32
Table 3-2. ADC Registers	32
Table 3-3. ADC firmware function	33
Table 3-4. Function adc_deinit	34
Table 3-5. Function adc_enable	35
Table 3-6. Function adc_disable	35
Table 3-7. Function adc_calibration_enable	36
Table 3-8. Function adc_dma_mode_enable	36
Table 3-9. Function adc_dma_mode_disable	37
Table 3-10. Function adc_tempsensor_enable	37
Table 3-11. Function adc_tempsensor_disable	38
Table 3-12. Function adc_vrefint_enable	38
Table 3-13. Function adc_vrefint_disable	39
Table 3-14. Function adc_discontinuous_mode_config	39
Table 3-15. Function adc_mode_config	40
Table 3-16. Function adc_special_function_config	41
Table 3-17. Function adc_data_alignment_config	42
Table 3-18. Function adc_channel_length_config	42
Table 3-19. Function adc_regular_channel_config	43
Table 3-20. Function adc_inserted_channel_config	44
Table 3-21. Function adc_inserted_channel_offset_config	45
Table 3-22. Function adc_external_trigger_config	46
Table 3-23. Function adc_external_trigger_source_config	47
Table 3-24. Function adc_software_trigger_enable	48
Table 3-25. Function adc_regular_data_read	48
Table 3-26. Function adc_inserted_data_read	49
Table 3-27. Function adc_sync_mode_convert_value_read	50
Table 3-28. Function adc_watchdog0_single_channel_enable	50
Table 3-29. Function adc_watchdog0_group_channel_enable	51
Table 3-30. Function adc_watchdog0_disable	51
Table 3-31. Function adc_watchdog1_channel_config	52
Table 3-32. Function adc_watchdog1_disable	53
Table 3-33. Function adc_watchdog0_threshold_config	53
Table 3-34. Function adc_watchdog1_threshold_config	54
Table 3-35. Function adc_resolution_config	54
Table 3-36. Function adc_oversample_mode_config	55
Table 3-37. Function adc_oversample_mode_enable	57
Table 3-38. Function adc_oversample_mode_disable	57

Table 3-39. Function adc_flag_get	58
Table 3-40. Function adc_flag_clear	59
Table 3-41. Function adc_interrupt_enable	59
Table 3-42. Function adc_interrupt_disable	60
Table 3-43. Function adc_interrupt_flag_get	61
Table 3-44. Function adc_interrupt_flag_clear	61
Table 3-45. BKP Registers	62
Table 3-46. BKP firmware function	62
Table 3-47. Enum bkp_data_register_enum	63
Table 3-48. Function bkp_deinit	63
Table 3-49. Function bkp_data_write	64
Table 3-50. Function bkp_data_read	65
Table 3-51. Function bkp_rtc_calibration_output_enable	65
Table 3-52. Function bkp_rtc_calibration_output_disable	66
Table 3-53. Function bkp_rtc_signal_output_enable	66
Table 3-54. Function bkp_rtc_signal_output_disable	67
Table 3-55. Function bkp_rtc_output_select	67
Table 3-56. Function bkp_rtc_clock_output_select	68
Table 3-57. Function bkp_rtc_clock_calibration_direction	68
Table 3-58. Function bkp_rtc_calibration_value_set	69
Table 3-59. bkp_osc32in_pin_select	69
Table 3-60. Function bkp_tamper_detection_enable	70
Table 3-61. Function bkp_tamper_detection_disable	70
Table 3-62. Function bkp_tamper_active_level_set	71
Table 3-63. Function bkp_tamper_interrupt_enable	71
Table 3-64. Function bkp_tamper_interrupt_disable	72
Table 3-65. Function bkp_flag_get	72
Table 3-66. Function bkp_flag_clear	73
Table 3-67. Function bkp_interrupt_flag_get	73
Table 3-68. Function bkp_interrupt_flag_clear	74
Table 3-69. CAN Registers	75
Table 3-70. CAN firmware function	76
Table 3-71. Structure can_error_counter_struct	77
Table 3-72. Structure can_parameter_struct	78
Table 3-73. Structure can_mailbox_descriptor_struct	78
Table 3-74. Structure can_rx_fifo_struct	79
Table 3-75. Structure can_fd_parameter_struct	79
Table 3-76. Structure can_rx_fifo_id_filter_struct	79
Table 3-77. Structure can_fifo_parameter_struct	80
Table 3-78. Structure can_pn_mode_filter_struct	80
Table 3-79. Structure can_pn_mode_config_struct	80
Table 3-80. Structure can_crc_struct	80
Table 3-81. Enum can_interrupt_enum	81
Table 3-82. Enum can_flag_enum	82

Table 3-83. Enum can_interrupt_flag_enum	84
Table 3-84. Enum can_operation_modes_enum	86
Table 3-85. Enum can_struct_type_enum	87
Table 3-86. Enum can_error_state_enum	87
Table 3-87. Function can_deinit	87
Table 3-88. Function can_software_reset	88
Table 3-89. Function can_init	88
Table 3-90. Function can_struct_para_init.....	89
Table 3-91. Function can_private_filter_config	90
Table 3-92. Function can_operation_mode_enter	90
Table 3-93. Function can_operation_mode_get	91
Table 3-94. Function can_inactive_mode_exit	92
Table 3-95. Function can_pn_mode_exit.....	92
Table 3-96. Function can_fd_config.....	93
Table 3-97. Function can_bitrate_switch_enable	93
Table 3-98. Function can_bitrate_switch_disable	94
Table 3-99. Function can_tdc_get	94
Table 3-100. Function can_tdc_enable	95
Table 3-101. Function can_tdc_disable	95
Table 3-102. Function can_rx_fifo_config	96
Table 3-103. Function can_rx_fifo_filter_table_config.....	97
Table 3-104. Function can_rx_fifo_read	97
Table 3-105. Function can_rx_fifo_filter_matching_number_get	98
Table 3-106. Function can_rx_fifo_clear	98
Table 3-107. Function can_ram_address_get.....	99
Table 3-108. Function can_mailbox_config	99
Table 3-109. Function can_mailbox_transmit_abort.....	100
Table 3-110. Function can_mailbox_transmit_inactive.....	101
Table 3-111. Function can_mailbox_receive_data_read	101
Table 3-112. Function can_mailbox_receive_lock.....	102
Table 3-113. Function can_mailbox_receive_unlock	103
Table 3-114. Function can_mailbox_receive_inactive	103
Table 3-115. Function can_mailbox_code_get	104
Table 3-116. Function can_error_counter_config	104
Table 3-117. Function can_error_counter_get.....	105
Table 3-118. Function can_error_state_get.....	106
Table 3-119. Function can_crc_get	106
Table 3-120. Function can_pn_mode_config	107
Table 3-121. Function can_pn_mode_filter_config	107
Table 3-122. Function can_pn_mode_num_of_match_get.....	108
Table 3-123. Function can_pn_mode_data_read.....	109
Table 3-124. Function can_self_reception_enable.....	109
Table 3-125. Function can_self_reception_disable.....	110
Table 3-126. Function can_transmit_abort_enable	110

Table 3-127. Function can_transmit_abort_disable	111
Table 3-128. Function can_auto_busoff_recovery_enable	111
Table 3-129. Function can_auto_busoff_recovery_disable.....	112
Table 3-130. Function can_time_sync_enable.....	112
Table 3-131. Function can_time_sync_disable.....	113
Table 3-132. Function can_edge_filter_mode_enable	113
Table 3-133. Function can_edge_filter_mode_disable	114
Table 3-134. Function can_ped_mode_enable	114
Table 3-135. Function can_ped_mode_disable	115
Table 3-136. Function can_arbitration_delay_bits_config.....	115
Table 3-137. Function can_bsp_mode_config	116
Table 3-138. Function can_flag_get.....	116
Table 3-139. Function can_flag_clear	117
Table 3-140. Function can_interrupt_enable	118
Table 3-141. Function can_interrupt_disable	118
Table 3-142. Function can_interrupt_flag_get	119
Table 3-143. Function can_interrupt_flag_clear	119
Table 3-144. CMP registers	120
Table 3-145. CMP firmware function	120
Table 3-146. Enum operating_mode_enum.....	121
Table 3-147. Enum inverting_input_enum.....	121
Table 3-148. Enum cmp_plus_input_enum	121
Table 3-149. Enum cmp_hysteresis_enum	122
Table 3-150. Enum cmp_output_enum	122
Table 3-151. Enum cmp_output_inv_enum.....	122
Table 3-152. Enum blanking_source_enum	122
Table 3-153. Enum cmp_output_state_enum	122
Table 3-154. Function cmp_deinit	123
Table 3-155. Function cmp_mode_init.....	123
Table 3-156. Function cmp_output_init.....	124
Table 3-157. Function cmp_outputblank_init.....	124
Table 3-158. Function cmp_enable	125
Table 3-159. Function cmp_disable	125
Table 3-160. Function cmp_voltage_scaler_enable	126
Table 3-161. Function cmp_voltage_scaler_disable	126
Table 3-162. Function cmp_scaler_bridge_enable.....	127
Table 3-163. Function cmp_scaler_bridge_disable.....	127
Table 3-164. Function cmp_lock_enable	127
Table 3-165. Function cmp_output_level_get.....	128
Table 3-166. CRC Registers	129
Table 3-167. CRC firmware function	129
Table 3-168. Function crc_deinit	129
Table 3-169. Function crc_reverse_output_data_enable.....	130
Table 3-170. Function crc_reverse_output_data_disable.....	130

Table 3-171. Function <code>crc_data_register_reset</code>	131
Table 3-172. Function <code>crc_data_register_read</code>	131
Table 3-173. Function <code>crc_free_data_register_read</code>	132
Table 3-174. Function <code>crc_free_data_register_write</code>	132
Table 3-175. Function <code>crc_init_data_register_write</code>	133
Table 3-176. Function <code>crc_input_data_reverse_config</code>	133
Table 3-177. Function <code>crc_polynomial_size_set</code>	134
Table 3-178. Function <code>crc_polynomial_set</code>	135
Table 3-179. Function <code>crc_single_data_calculate</code>	135
Table 3-180. Function <code>crc_block_data_calculate</code>	136
Table 3-181. DBG Registers	137
Table 3-182. DBG firmware function	137
Table 3-183. Enum <code>dbg_periph_enum</code>	137
Table 3-184. Function <code>dbg_deinit</code>	138
Table 3-185. Function <code>dbg_id_get</code>	138
Table 3-186. Function <code>dbg_low_power_enable</code>	139
Table 3-187. Function <code>dbg_low_power_disable</code>	139
Table 3-188. Function <code>dbg_periph_enable</code>	140
Table 3-189. Function <code>dbg_periph_disable</code>	141
Table 3-190. DAC Registers	141
Table 3-191. DAC firmware function	142
Table 3-192. Function <code>dac_deinit</code>	142
Table 3-193. Function <code>dac_enable</code>	143
Table 3-194. Function <code>dac_disable</code>	143
Table 3-195. Function <code>dac_dma_enable</code>	144
Table 3-196. Function <code>dac_dma_disable</code>	144
Table 3-197. Function <code>dac_output_buffer_enable</code>	145
Table 3-198. Function <code>dac_output_buffer_disable</code>	145
Table 3-199. Function <code>dac_output_value_get</code>	146
Table 3-200. Function <code>dac_data_set</code>	146
Table 3-201. Function <code>dac_trigger_enable</code>	147
Table 3-202. Function <code>dac_trigger_disable</code>	147
Table 3-203. Function <code>dac_trigger_source_config</code>	148
Table 3-204. Function <code>dac_software_trigger_enable</code>	149
Table 3-205. Function <code>dac_software_trigger_disable</code>	149
Table 3-206. Function <code>dac_wave_mode_config</code>	150
Table 3-207. Function <code>dac_wave_bit_width_config</code>	150
Table 3-208. Function <code>dac_lfsr_noise_config</code>	151
Table 3-209. Function <code>dac_triangle_noise_config</code>	151
Table 3-210. Function <code>dac_flag_get</code>	152
Table 3-211. Function <code>dac_flag_clear</code>	152
Table 3-212. Function <code>dac_interrupt_enable</code>	153
Table 3-213. Function <code>dac_interrupt_disable</code>	153
Table 3-214. Function <code>dac_interrupt_flag_get</code>	154

Table 3-215. Function dac_interrupt_flag_clear	155
Table 3-216. DMA Registers	155
Table 3-217. DMAMUX Registers	156
Table 3-218. DMA firmware function	156
Table 3-219. DMAMUX firmware function	157
Table 3-220. Structure dma_parameter_struct	158
Table 3-221. Structure dmamux_sync_parameter_struct	158
Table 3-222. Structure dmamux_gen_parameter_struct	158
Table 3-223. Enum dma_channel_enum	159
Table 3-224. Enum dmamux_multiplexer_channel_enum	159
Table 3-225. Enum dmamux_generator_channel_enum	160
Table 3-226. Enum dmamux_interrupt_enum	160
Table 3-227. Enum dmamux_flag_enum	161
Table 3-228. Enum dmamux_interrupt_flag_enum	161
Table 3-229. Function dma_deinit	162
Table 3-230. Function dma_para_init	163
Table 3-231. Function dma_init	163
Table 3-232. Function dma_circulation_enable	164
Table 3-233. Function dma_circulation_disable	165
Table 3-234. Function dma_memory_to_memory_enable	165
Table 3-235. Function dma_memory_to_memory_disable	166
Table 3-236. Function dma_channel_enable	166
Table 3-237. Function dma_channel_disable	167
Table 3-238. Function dma_periph_address_config	167
Table 3-239. Function dma_memory_address_config	168
Table 3-240. Function dma_transfer_number_config	168
Table 3-241. Function dma_transfer_number_get	169
Table 3-242. Function dma_priority_config	170
Table 3-243. Function dma_memory_width_config	170
Table 3-244. Function dma_periph_width_config	171
Table 3-245. Function dma_memory_increase_enable	172
Table 3-246. Function dma_memory_increase_disable	172
Table 3-247. Function dma_periph_increase_enable	173
Table 3-248. Function dma_periph_increase_disable	173
Table 3-249. Function dma_transfer_direction_config	174
Table 3-250. Function dma_flag_get	174
Table 3-251. Function dma_flag_clear	175
Table 3-252. Function dma_interrupt_enable	176
Table 3-253. Function dma_interrupt_disable	176
Table 3-254. Function dma_interrupt_flag_get	177
Table 3-255. Function dma_interrupt_flag_clear	178
Table 3-256. Function dmamux_sync_struct_para_init	178
Table 3-257. Function dmamux_synchronization_init	179
Table 3-258. Function dmamux_synchronization_enable	180

Table 3-259. Function dmamux_synchronization_disable	180
Table 3-260. Function dmamux_event_generation_enable	181
Table 3-261. Function dmamux_event_generation_disable	181
Table 3-262. Function dmamux_gen_struct_para_init	182
Table 3-263. Function dmamux_request_generator_init	182
Table 3-264. Function dmamux_request_generator_channel_enable	183
Table 3-265. Function dmamux_request_generator_channel_disable	184
Table 3-266. Function dmamux_synchronization_polarity_config	184
Table 3-267. Function dmamux_request_forward_number_config	185
Table 3-268. Function dmamux_sync_id_config	186
Table 3-269. Function dmamux_request_id_config	187
Table 3-270. Function dmamux_trigger_polarity_config	191
Table 3-271. Function dmamux_request_generate_number_config	192
Table 3-272. Function dmamux_trigger_id_config	193
Table 3-273. Function dmamux_flag_get	194
Table 3-274. Function dmamux_flag_clear	196
Table 3-275. Function dmamux_interrupt_enable	197
Table 3-276. Function dmamux_interrupt_disable	198
Table 3-277. Function dmamux_interrupt_flag_get	199
Table 3-278. Function dmamux_interrupt_flag_clear	201
Table 3-279. EXTI Registers	202
Table 3-280. EXTI firmware function	202
Table 3-281. exti_line_enum	203
Table 3-282. exti_mode_enum	203
Table 3-283. exti_trig_type_enum	204
Table 3-284. Function exti_deinit	204
Table 3-285. Function exti_init	204
Table 3-286. Function exti_interrupt_enable	205
Table 3-287. Function exti_interrupt_disable	205
Table 3-288. Function exti_event_enable	206
Table 3-289. Function exti_event_disable	206
Table 3-290. Function exti_software_interrupt_enable	207
Table 3-291. Function exti_software_interrupt_disable	207
Table 3-292. Function exti_flag_get	208
Table 3-293. Function exti_flag_clear	208
Table 3-294. Function exti_interrupt_flag_get	209
Table 3-295. Function exti_interrupt_flag_clear	209
Table 3-296. FMC Registers	210
Table 3-297. FMC firmware function	210
Table 3-298. fmc_state_enum	212
Table 3-299. fmc_sram_mode_enum	213
Table 3-300. fmc_area_enum	213
Table 3-301. fmc_flag_enum	213
Table 3-302. fmc_interrupt_flag_enum	215

Table 3-303. fmc_interrupt_enum	216
Table 3-304. Function fmc_unlock	216
Table 3-305. Function fmc_bank0_unlock.....	216
Table 3-306. Function fmc_bank1_unlock.....	217
Table 3-307. Function fmc_lock	217
Table 3-308. Function fmc_bank0_lock	218
Table 3-309. Function fmc_bank1_lock	218
Table 3-310. Function fmc_wsnt_set	219
Table 3-311. Function fmc_prefetch_enable	219
Table 3-312. Function fmc_prefetch_disable	220
Table 3-313. Function fmc_cache_enable	220
Table 3-314. Function fmc_cache_disable	221
Table 3-315. Function fmc_cache_reset_enable	221
Table 3-316. Function fmc_cache_reset_disable	222
Table 3-317. Function fmc_powerdown_mode_set.....	222
Table 3-318. Function fmc_sleep_mode_set	223
Table 3-319. Function fmc_sram_mode_config	223
Table 3-320. Function fmc_sram_mode_get.....	224
Table 3-321. Function fmc_blank_check	225
Table 3-322. Function fmc_page_erase.....	225
Table 3-323. Function fmc_bank0_mass_erase	226
Table 3-324. Function fmc_bank1_mass_erase	227
Table 3-325. Function fmc_dflash_mass_erase	228
Table 3-326. Function fmc_mass_erase	228
Table 3-327. Function fmc_doubleword_program	229
Table 3-328. Function fmc_fast_program.....	230
Table 3-329. Function otp_doubleword_program	231
Table 3-330. Function eeprom_word_program.....	232
Table 3-331. Function eeprom_word_read.....	233
Table 3-332. Function ob_unlock	233
Table 3-333. Function ob_lock	234
Table 3-334. Function ob_reset	234
Table 3-335. Function ob_erase	235
Table 3-336. Function ob_write_protection_enable	236
Table 3-337. Function ob_security_protection_config	237
Table 3-338. Function ob_user_write.....	237
Table 3-339. Function ob_data_program.....	239
Table 3-340. Function ob_user_get.....	239
Table 3-341. Function ob_data_get.....	240
Table 3-342. Function ob_write_protection_get	240
Table 3-343. Function ob_bk1_write_protection_get.....	241
Table 3-344. Function ob_df_write_protection_get.....	241
Table 3-345. Function ob_ep_write_protection_get.....	242
Table 3-346. Function ob_plevel_get.....	242

Table 3-347. Function ob1_lock_config.....	243
Table 3-348. Function ob1_epload_config	244
Table 3-349. Function ob1_eeprom_parameter_config	245
Table 3-350. Function dflash_size_get	246
Table 3-351. Function eeprom_backup_size_get.....	247
Table 3-352. Function eeprom_size_get.....	247
Table 3-353. Function fmc_flag_get.....	248
Table 3-354. Function fmc_flag_clear.....	248
Table 3-355. Function fmc_interrupt_enable	250
Table 3-356. Function fmc_interrupt_disable	250
Table 3-357. Function fmc_interrupt_flag_get.....	251
Table 3-358. Function fmc_interrupt_flag_clear.....	251
Table 3-359. FWDGT Registers.....	252
Table 3-360. FWDGT firmware function.....	252
Table 3-361. Function fwdgt_write_ensable	252
Table 3-362. Function fwdgt_write_disable	253
Table 3-363. Function fwdgt_enable	253
Table 3-364. Function fwdgt_prescaler_value_config	254
Table 3-365. Function fwdgt_reload_value_config	254
Table 3-366. Function fwdgt_window_value_config	255
Table 3-367. Function fwdgt_counter_reload	255
Table 3-368. Function fwdgt_config.....	256
Table 3-369. Function fwdgt_flag_get.....	257
Table 3-370. GPIO Registers	257
Table 3-371. GPIO firmware function	258
Table 3-372. Function gpio_deinit.....	258
Table 3-373. Function gpio_mode_set.....	259
Table 3-374. Function gpio_output_options_set.....	260
Table 3-375. Function gpio_bit_set.....	261
Table 3-376. Function gpio_bit_reset	261
Table 3-377. Function gpio_bit_write.....	262
Table 3-378. Function gpio_port_write	263
Table 3-379. Function gpio_input_bit_get.....	263
Table 3-380. Function gpio_input_port_get	264
Table 3-381. Function gpio_output_bit_get	264
Table 3-382. Function gpio_output_port_get.....	265
Table 3-383. Function gpio_af_set.....	265
Table 3-384. Function gpio_pin_lock.....	266
Table 3-385. Function gpio_bit_toggle	267
Table 3-386. Function gpio_port_toggle.....	268
Table 3-387. I2C Registers	268
Table 3-388. I2C firmware function	269
Table 3-389. i2c_interrupt_flag_enum	270
Table 3-390. Function i2c_deinit	271

Table 3-391. Function i2c_timing_config	271
Table 3-392. Function i2c_digital_noise_filter_config	272
Table 3-393. Function i2c_analog_noise_filter_enable.....	273
Table 3-394. Function i2c_analog_noise_filter_disable.....	274
Table 3-395. Function i2c_master_clock_config	274
Table 3-396. Function i2c_master_addressing	275
Table 3-397. Function i2c_address10_header_enable	275
Table 3-398. Function i2c_address10_header_disable	276
Table 3-399. Function i2c_address10_enable.....	276
Table 3-400. Function i2c_address10_disable.....	277
Table 3-401. Function i2c_automatic_end_enable	277
Table 3-402. Function i2c_automatic_end_disable	278
Table 3-403. Function i2c_slave_response_to_gcall_enable.....	278
Table 3-404. Function i2c_slave_response_to_gcall_disable.....	279
Table 3-405. Function i2c_stretch_scl_low_enable	279
Table 3-406. Function i2c_stretch_scl_low_disable	280
Table 3-407. Function i2c_address_config	280
Table 3-408. Function i2c_address_bit_compare_config.....	281
Table 3-409. Function i2c_address_disable.....	282
Table 3-410. Function i2c_second_address_config.....	283
Table 3-411. Function i2c_second_address_disable	284
Table 3-412. Function i2c_receved_address_get	284
Table 3-413. Function i2c_slave_byte_control_enable.....	285
Table 3-414. Function i2c_slave_byte_control_disable	285
Table 3-415. Function i2c_nack_enable	286
Table 3-416. Function i2c_nack_disable	286
Table 3-417. Function i2c_enable.....	287
Table 3-418. Function i2c_disable.....	287
Table 3-419. Function i2c_start_on_bus	288
Table 3-420. Function i2c_stop_on_bus.....	288
Table 3-421. Function i2c_data_transmit	289
Table 3-422. Function i2c_data_receive	289
Table 3-423. Function i2c_reload_enable.....	290
Table 3-424. Function i2c_reload_disable.....	290
Table 3-425. Function i2c_transfer_byte_number_config	291
Table 3-426. Function i2c_dma_enable	291
Table 3-427. Function i2c_dma_disable	292
Table 3-428. Function i2c_pec_transfer	293
Table 3-429. Function i2c_pec_enable	293
Table 3-430. Function i2c_pec_disable	294
Table 3-431. Function i2c_pec_value_get	294
Table 3-432. Function i2c_smbus_alert_enable	295
Table 3-433. Function i2c_smbus_alert_disable	295
Table 3-434. Function i2c_smbus_default_addr_enable	296

Table 3-435. Function i2c_smbus_default_addr_disable	296
Table 3-436. Function i2c_smbus_host_addr_enable	297
Table 3-437. Function i2c_smbus_host_addr_disable	297
Table 3-438. Function i2c_extented_clock_timeout_enable	298
Table 3-439. Function i2c_extented_clock_timeout_disable	298
Table 3-440. Function i2c_clock_timeout_enable	299
Table 3-441. Function i2c_clock_timeout_disable	299
Table 3-442. Function i2c_bus_timeout_b_config	300
Table 3-443. Function i2c_bus_timeout_a_config.....	300
Table 3-444. Function i2c_idle_clock_timeout_config.....	301
Table 3-445. Function i2c_flag_get	301
Table 3-446. Function i2c_flag_clear	302
Table 3-447. Function i2c_interrupt_enable.....	303
Table 3-448. Function i2c_interrupt_disable.....	304
Table 3-449. Function i2c_interrupt_flag_get	305
Table 3-450. Function i2c_interrupt_flag_clear	306
Table 3-451. MFCOM Registers	307
Table 3-452. MFCOM firmware function	307
Table 3-453. Struct mfcom_timer_parameter_struct.....	308
Table 3-454. Struct mfcom_shifter_parameter_struct	309
Table 3-455. Function mfcom_deinit.....	309
Table 3-456. Function mfcom_software_reset.....	310
Table 3-457. Function mfcom_enable.....	310
Table 3-458. Function mfcom_disable	311
Table 3-459. Function mfcom_timer_struct_para_init	311
Table 3-460. Function mfcom_shifter_struct_para_init	312
Table 3-461. Function mfcom_timer_init	312
Table 3-462. Function mfcom_shifter_init.....	313
Table 3-463. Function mfcom_timer_pin_config	314
Table 3-464. Function mfcom_shifter_pin_config.....	315
Table 3-465. Function mfcom_timer_enable	316
Table 3-466. Function mfcom_shifter_enable.....	317
Table 3-467. Function mfcom_timer_disable	317
Table 3-468. Function mfcom_shifter_disable.....	318
Table 3-469. Function mfcom_timer_cmpvalue_set	318
Table 3-470. Function mfcom_timer_cmpvalue_get	319
Table 3-471. Function mfcom_timer_dismode_set	320
Table 3-472. Function mfcom_shifter_stopbit_set.....	320
Table 3-473. Function mfcom_buffer_write.....	321
Table 3-474. Function mfcom_buffer_read	322
Table 3-475. Function mfcom_shifter_flag_get	323
Table 3-476. Function mfcom_shifter_error_flag_get.....	323
Table 3-477. Function mfcom_timer_flag_get	324
Table 3-478. Function mfcom_shifter_interrupt_flag_get	324

Table 3-479. Function mfcom_shifter_error_interrupt_flag_get	325
Table 3-480. Function mfcom_timer_interrupt_flag_get.....	325
Table 3-481. Function mfcom_shifter_flag_clear	326
Table 3-482. Function mfcom_shifter_error_flag_clear	326
Table 3-483. Function mfcom_timer_flag_clear	327
Table 3-484. Function mfcom_shifter_interrupt_enable.....	327
Table 3-485. Function mfcom_shifter_error_interrupt_enable	328
Table 3-486. Function mfcom_timer_interrupt_enable	328
Table 3-487. Function mfcom_shifter_dma_enable	329
Table 3-488. Function mfcom_shifter_interrupt_disable.....	329
Table 3-489. Function mfcom_shifter_error_interrupt_disable	330
Table 3-490. Function mfcom_timer_interrupt_disable	330
Table 3-491. Function mfcom_shifter_dma_disable	331
Table 3-492. NVIC Registers	331
Table 3-493. SysTick Registers	332
Table 3-494. MISC firmware function	332
Table 3-495. IRQn_Type.....	332
Table 3-496. Function nvic_priority_group_set.....	334
Table 3-497. Function nvic_irq_enable.....	335
Table 3-498. Function nvic_irq_disable.....	335
Table 3-499. Function nvic_system_reset.....	336
Table 3-500. Function nvic_vector_table_set	336
Table 3-501. Function system_lowpower_set.....	337
Table 3-502. Function system_lowpower_reset	338
Table 3-503. Function systick_clksource_set.....	338
Table 3-504. PMU Registers	339
Table 3-505. PMU firmware function	339
Table 3-506. Function pmu_deinit.....	340
Table 3-507. Function pmu_lvd_select	340
Table 3-508. Function pmu_lvd_disable.....	341
Table 3-509. Function pmu_ovd_select.....	341
Table 3-510. Function pmu_ovd_disable.....	342
Table 3-511. Function pmu_lowdriver_mode_enable	342
Table 3-512. Function pmu_lowdriver_mode_disable	343
Table 3-513. Function pmu_sram1_poweroff_mode_enable	343
Table 3-514. Function pmu_sram1_poweroff_mode_disable	344
Table 3-515. Function pmu_sram2_poweroff_mode_enable	344
Table 3-516. Function pmu_sram2_poweroff_mode_disable	345
Table 3-517. Function pmu_to_sleepmode	345
Table 3-518. Function pmu_to_deepsleepmode.....	346
Table 3-519. Function pmu_to_standbymode	347
Table 3-520. Function pmu_wakeup_pin_enable	347
Table 3-521. Function pmu_wakeup_pin_disable	348
Table 3-522. Function pmu_backup_write_enable.....	348

Table 3-523. Function pmu_backup_write_disable	349
Table 3-524. Function pmu_flag_get.....	349
Table 3-525. Function pmu_flag_clear.....	350
Table 3-526. RCU Registers	351
Table 3-527. RCU firmware function	351
Table 3-528. Enum rcu_periph_enum	352
Table 3-529. Enum rcu_periph_sleep_enum.....	353
Table 3-530. Enum rcu_periph_reset_enum	354
Table 3-531. Enum rcu_flag_enum.....	355
Table 3-532. Enum rcu_int_flag_enum	355
Table 3-533. Enum rcu_int_flag_clear_enum.....	356
Table 3-534. Enum rcu_int_enum.....	356
Table 3-535. Enum rcu_osci_type_enum	357
Table 3-536. Enum rcu_clock_freq_enum	357
Table 3-537. Function rcu_deinit.....	357
Table 3-538. Function rcu_periph_clock_enable.....	358
Table 3-539. Function rcu_periph_clock_disable.....	359
Table 3-540. Function rcu_periph_reset_enable	359
Table 3-541. Function rcu_periph_reset_disable	360
Table 3-542. Function rcu_periph_clock_sleep_enable	361
Table 3-543. Function rcu_periph_clock_sleep_disable	362
Table 3-544. Function rcu_bkp_reset_enable	362
Table 3-545. Function rcu_bkp_reset_disable	363
Table 3-546. Function rcu_system_clock_source_config	363
Table 3-547. Function rcu_system_clock_source_get	364
Table 3-548. Function rcu_ahb_clock_config	364
Table 3-549. Function rcu_apb1_clock_config	365
Table 3-550. Function rcu_apb2_clock_config	365
Table 3-551. Function rcu_ckout_config	366
Table 3-552. Function rcu_pll_config	367
Table 3-553. Function rcu_double_pll_enable	367
Table 3-554. Function rcu_double_pll_disable	368
Table 3-555. Function rcu_system_reset_enable.....	368
Table 3-556. Function rcu_system_reset_disable	369
Table 3-557. Function rcu_adc_clock_config	370
Table 3-558. Function rcu_rtc_clock_config.....	370
Table 3-559. Function rcu_usart_clock_config	371
Table 3-560. Function rcu_can_clock_config	372
Table 3-561. Function rcu_lxtal_drive_capability_config.....	372
Table 3-562. Function rcu_osci_stab_wait.....	373
Table 3-563. Function rcu_osci_on	374
Table 3-564. Function rcu_osci_off.....	374
Table 3-565. Function rcu_osci_bypass_mode_enable.....	375
Table 3-566. Function rcu_osci_bypass_mode_disable.....	375

Table 3-567. Function	376
Table 3-568. Function rcu_hxtal_prediv_config	376
Table 3-569. Function rcu_irc8m_adjust_value_set.....	377
Table 3-570. Function rcu_hxtal_clock_monitor_enable	377
Table 3-571. Function rcu_hxtal_clock_monitor_disable.....	378
Table 3-572. Function rcu_lxtal_clock_monitor_enable	378
Table 3-573. Function rcu_lxtal_clock_monitor_disable	379
Table 3-574. Function rcu_voltage_key_unlock	379
Table 3-575. Function rcu_deepsleep_voltage_set.....	380
Table 3-576. Function rcu_clock_freq_get.....	381
Table 3-577. Function rcu_flag_get.....	381
Table 3-578. Function rcu_all_reset_flag_clear	382
Table 3-579. Function rcu_interrupt_flag_get.....	383
Table 3-580. Function rcu_interrupt_flag_clear	384
Table 3-581. Function rcu_interrupt_enable	385
Table 3-582. Function rcu_interrupt_disable	385
Table 3-583. RTC Registers	386
Table 3-584. RTC firmware function.....	387
Table 3-585. Function rtc_configuration_mode_enter	387
Table 3-586. Function rtc_configuration_mode_exit	388
Table 3-587. Function rtc_lwoff_wait	388
Table 3-588. Function rtc_register_sync_wait	389
Table 3-589. Function rtc_counter_get.....	389
Table 3-590. Function rtc_counter_set	390
Table 3-591. Function rtc_prescaler_set	390
Table 3-592. Function rtc_alarm_config	391
Table 3-593. Function rtc_divider_get	391
Table 3-594. Function rtc_interrupt_enable	392
Table 3-595. Function rtc_interrupt_disable	392
Table 3-596. Function rtc_flag_get.....	393
Table 3-597. Function rtc_flag_clear.....	394
Table 3-598. Function rtc_interrupt_flag_get.....	394
Table 3-599. Function rtc_interrupt_flag_clear.....	395
Table 3-600. SPI/I2S Registers.....	396
Table 3-601. SPI/I2S firmware function.....	396
Table 3-602. spi_parameter_struct.....	397
Table 3-603. Function spi_i2s_deinit	398
Table 3-604. Function spi_struct_para_init.....	398
Table 3-605. Function spi_init	399
Table 3-606. Function spi_enable	400
Table 3-607. Function spi_disable	400
Table 3-608. Function i2s_init.....	401
Table 3-609. Function i2s_psc_config	402
Table 3-610. Function i2s_enable.....	403

Table 3-611. Function i2s_disable	404
Table 3-612. Function spi_nss_output_enable	404
Table 3-613. Function spi_nss_output_disable	405
Table 3-614. Function spi_nss_internal_high	405
Table 3-615. Function spi_nss_internal_low	406
Table 3-616. Function spi_dma_enable	406
Table 3-617. Function spi_dma_disable	407
Table 3-618. Function spi_i2s_data_frame_format_config	408
Table 3-619. Function spi_i2s_data_transmit	408
Table 3-620. Function spi_i2s_data_receive	409
Table 3-621. Function spi_bidirectional_transfer_config	409
Table 3-622. Function spi_i2s_format_error_clear	410
Table 3-623. Function spi_crc_polynomial_set	410
Table 3-624. Function spi_crc_polynomial_get	411
Table 3-625. Function spi_crc_on	412
Table 3-626. Function spi_crc_off	412
Table 3-627. Function spi_crc_next	413
Table 3-628. Function spi_crc_get	413
Table 3-629. Function spi_crc_error_clear	414
Table 3-630. Function spi_ti_mode_enable	414
Table 3-631. Function spi_ti_mode_disable	415
Table 3-632. Function spi_nssp_mode_enable	415
Table 3-633. Function spi_nssp_mode_disable	416
Table 3-634. Function spi_quad_enable	416
Table 3-635. Function spi_quad_disable	417
Table 3-636. Function spi_quad_write_enable	417
Table 3-637. Function spi_quad_read_enable	418
Table 3-638. Function spi_quad_io23_output_enable	418
Table 3-639. Function spi_quad_io23_output_disable	419
Table 3-640. Function spi_i2s_interrupt_enable	419
Table 3-641. Function spi_i2s_interrupt_disable	420
Table 3-642. Function spi_i2s_interrupt_flag_get	420
Table 3-643. Function spi_i2s_flag_get	421
Table 3-644. SYSCFG Registers	422
Table 3-645. SYSCFG firmware function	423
Table 3-646. Function syscfg_deinit	423
Table 3-647. Function syscfg_exti_line_config	424
Table 3-648. Function syscfg_pin_remap_enable	424
Table 3-649. Function syscfg_pin_remap_disable	425
Table 3-650. Function syscfg_adc_ch_remap_config	426
Table 3-651. Function syscfg_timer_eti_sel	426
Table 3-652. Function syscfg_timer_bkin_select_trigsel	427
Table 3-653. Function syscfg_timer_bkin_select_gpio	427
Table 3-654. Function syscfg_timer7_ch0n_select	428

Table 3-655. Function syscfg_lock_config.....	428
Table 3-656. Function syscfg_flag_get.....	429
Table 3-657. Function syscfg_flag_clear.....	430
Table 3-658. Function syscfg_interrupt_enable	430
Table 3-659. Function syscfg_interrupt_disable	431
Table 3-660. Function syscfg_interrupt_flag_get.....	432
Table 3-661. Function syscfg_sram_ecc_address_get.....	433
Table 3-662. Function syscfg_sram_ecc_bit_get	433
Table 3-663. TIMERx Registers.....	434
Table 3-664. TIMERx firmware function.....	435
Table 3-665. Structure timer_parameter_struct.....	438
Table 3-666. Structure timer_break_parameter_struct	438
Table 3-667. Structure timer_oc_parameter_struct.....	439
Table 3-668. Structure timer_omc_parameter_struct	439
Table 3-669. Structure timer_ic_parameter_struct.....	439
Table 3-670. Structure timer_break_ext_input_struct.....	440
Table 3-671. Structure timer_free_complementary_parameter_struct	440
Table 3-672. Function timer_deinit.....	440
Table 3-673. Function timer_struct_para_init	441
Table 3-674. Function timer_init.....	441
Table 3-675. Function timer_enable.....	442
Table 3-676. Function timer_disable.....	443
Table 3-677. Function timer_auto_reload_shadow_enable	443
Table 3-678. Function timer_auto_reload_shadow_disable.....	444
Table 3-679. Function timer_update_event_enable	444
Table 3-680. Function timer_update_event_disable	445
Table 3-681. Function timer_counter_alignment	445
Table 3-682. Function timer_counter_up_direction	446
Table 3-683. timer_counter_down_direction	447
Table 3-684. Function timer_prescaler_config	447
Table 3-685. Function timer_repetition_value_config.....	448
Table 3-686. Function timer_autoreload_value_config.....	449
Table 3-687. Function timer_counter_value_config.....	449
Table 3-688. Function timer_counter_read	450
Table 3-689. Function timer_prescaler_read	450
Table 3-690. Function timer_single_pulse_mode_config.....	451
Table 3-691. Function timer_update_source_config.....	452
Table 3-692. Function timer_channel_control_shadow_config	452
Table 3-693. Function timer_channel_control_shadow_update_config	453
Table 3-694. Function timer_dma_enable	454
Table 3-695. Function timer_dma_disable	454
Table 3-696. Function timer_channel_dma_request_source_select.....	455
Table 3-697. Function timer_dma_transfer_config	456
Table 3-698. Function timer_event_software_generate	458

Table 3-699. Function timer_break_struct_para_init	460
Table 3-700. Function timer_break_config.....	460
Table 3-701. Function timer_break_enable	461
Table 3-702. Function timer_break_disable	462
Table 3-703. Function timer_automatic_output_enable	462
Table 3-704. Function timer_automatic_output_disable	463
Table 3-705. Function timer_primary_output_config.....	463
Table 3-706. Function timer_channel_output_struct_para_init	464
Table 3-707. Function timer_channel_output_config	464
Table 3-708. Function timer_channel_output_mode_config.....	465
Table 3-709. Function timer_channel_output_pulse_value_config.....	467
Table 3-710. Function timer_channel_output_shadow_config	467
Table 3-711. Function timer_channel_output_clear_config	468
Table 3-712. Function timer_channel_output_polarity_config	469
Table 3-713. Function timer_channel_complementary_output_polarity_config.....	470
Table 3-714. Function timer_channel_output_state_config	471
Table 3-715. Function timer_channel_complementary_output_state_config	472
Table 3-716. Function timer_channel_input_struct_para_init.....	473
Table 3-717. Function timer_input_capture_config	474
Table 3-718. Function timer_channel_input_capture_prescaler_config.....	475
Table 3-719. Function timer_channel_capture_value_register_read	476
Table 3-720. Function timer_input_pwm_capture_config	476
Table 3-721. Function timer_hall_mode_config	477
Table 3-722. Function timer_multi_mode_channel_output_parameter_struct_init.....	478
Table 3-723. Function timer_multi_mode_channel_output_config	478
Table 3-724. Function timer_multi_mode_channel_mode_config.....	479
Table 3-725. Function timer_input_trigger_source_select.....	480
Table 3-726. Function timer_master_output_trigger_source_select	481
Table 3-727. Function timer_slave_mode_select	482
Table 3-728. Function timer_master_slave_mode_config.....	483
Table 3-729. Function timer_external_trigger_config	484
Table 3-730. Function timer_quadrature_decoder_mode_config.....	485
Table 3-731. Function timer_internal_clock_config	486
Table 3-732. Function timer_internal_trigger_as_external_clock_config.....	487
Table 3-733. Function timer_external_trigger_as_external_clock_config.....	487
Table 3-734. Function timer_external_clock_mode0_config.....	488
Table 3-735. Function timer_external_clock_mode1_config.....	489
Table 3-736. Function timer_external_clock_mode1_disable	490
Table 3-737. Function timer_channel_remap_config.....	491
Table 3-738. Function timer_write_chxval_register_config	491
Table 3-739. Function timer_output_value_selection_config	492
Table 3-740. Function timer_output_match_pulse_select.....	493
Table 3-741. Function timer_channel_composite_pwm_mode_config.....	494
Table 3-742. Function timer_channel_composite_pwm_mode_output_pulse_value_config.....	494

Table 3-743. Function timer_channel_additional_compare_value_config	495
Table 3-744. Function timer_channel_additional_output_shadow_config	496
Table 3-745. Function timer_break_external_input_struct_para_init	497
Table 3-746. Function timer_break_external_input_config	497
Table 3-747. Function timer_break_external_input_enable	498
Table 3-748. Function timer_break_external_input_disable	499
Table 3-749. Function timer_break_external_input_polarity_config	500
Table 3-750. Function timer_channel_break_control_config	501
Table 3-751. Function timer_channel_dead_time_config	502
Table 3-752. Function timer_free_complementary_struct_para_init	502
Table 3-753. Function timer_channel_free_complementary_config	503
Table 3-754. Function timer_flag_get	504
Table 3-755. Function timer_flag_clear	505
Table 3-756. Function timer_interrupt_enable	506
Table 3-757. Function timer_interrupt_disable	508
Table 3-758. Function timer_interrupt_flag_get	509
Table 3-759. Function timer_interrupt_flag_clear	510
Table 3-760. TRIGSEL Registers	511
Table 3-761. TRIGSEL firmware function	512
Table 3-762. Enum trigselsource_enum	512
Table 3-763. Enum trigselperiph_enum	514
Table 3-764. Function trigsels_init	515
Table 3-765. Function trigsels_trigger_source_get	516
Table 3-766. Function trigsels_register_lock_set	516
Table 3-767. Function trigsels_register_lock_get	517
Table 3-768. USART Registers	517
Table 3-769. USART firmware function	518
Table 3-770. Enum usart_flag_enum	520
Table 3-771. Enum usart_interrupt_flag_enum	521
Table 3-772. Enum usart_interrupt_enum	521
Table 3-773. Enum usart_invert_enum	522
Table 3-774. Function usart_deinit	522
Table 3-775. Function usart_baudrate_set	522
Table 3-776. Function usart_parity_config	523
Table 3-777. Function usart_word_length_set	524
Table 3-778. Function usart_stop_bit_set	524
Table 3-779. Function usart_enable	525
Table 3-780. Function usart_disable	525
Table 3-781. Function usart_transmit_config	526
Table 3-782. Function usart_receive_config	527
Table 3-783. Function usart_data_first_config	527
Table 3-784. Function usart_invert_config	528
Table 3-785. Function usart_overrun_enable	528
Table 3-786. Function usart_overrun_disable	529

Table 3-787. Function usart_oversample_config	529
Table 3-788. Function usart_sample_bit_config	530
Table 3-789. Function usart_receiver_timeout_enable	531
Table 3-790. Function usart_receiver_timeout_disable	531
Table 3-791. Function usart_receiver_timeout_threshold_config	532
Table 3-792. Function usart_data_transmit	532
Table 3-793. Function usart_data_receive	533
Table 3-794. Function usart_command_enable	533
Table 3-795. Function usart_address_config	534
Table 3-796. Function usart_address_detection_mode_config	535
Table 3-797. Function usart_mute_mode_enable	535
Table 3-798. Function usart_mute_mode_disable	536
Table 3-799. Function usart_mute_mode_wakeup_config	536
Table 3-800. Function usart_lin_mode_enable	537
Table 3-801. Function usart_lin_mode_disable	537
Table 3-802. Function usart_lin_break_detection_length_config	538
Table 3-803. Function usart_halfduplex_enable	538
Table 3-804. Function usart_halfduplex_disable	539
Table 3-805. Function usart_clock_enable	540
Table 3-806. Function usart_clock_disable	540
Table 3-807. Function usart_synchronous_clock_config	541
Table 3-808. Function usart_guard_time_config	541
Table 3-809. Function usart_smartcard_mode_enable	542
Table 3-810. Function usart_smartcard_mode_disable	542
Table 3-811. Function usart_smartcard_mode_nack_enable	543
Table 3-812. Function usart_smartcard_mode_nack_disable	543
Table 3-813. Function usart_smartcard_mode_early_nack_enable	544
Table 3-814. Function usart_smartcard_mode_early_nack_disable	544
Table 3-815. Function usart_smartcard_autoretry_config	545
Table 3-816. Function usart_block_length_config	546
Table 3-817. Function usart_irda_mode_enable	546
Table 3-818. Function usart_irda_mode_disable	547
Table 3-819. Function usart_prescaler_config	547
Table 3-820. Function usart_irda_lowpower_config	548
Table 3-821. Function usart_hardware_flow_rts_config	548
Table 3-822. Function usart_hardware_flow_cts_config	549
Table 3-823. Function usart_hardware_flow_coherence_config	550
Table 3-824. Function usart_rs485_driver_enable	550
Table 3-825. Function usart_rs485_driver_disable	551
Table 3-826. Function usart_driver_asserttime_config	551
Table 3-827. Function usart_driver_deasserttime_config	552
Table 3-828. Function usart_depolarity_config	552
Table 3-829. Function usart_dma_receive_config	553
Table 3-830. Function usart_dma_transmit_config	554

Table 3-831. Function usart_reception_error_dma_disable	554
Table 3-832. Function usart_reception_error_dma_enable	555
Table 3-833. Function usart_wakeup_enable	555
Table 3-834. Function usart_wakeup_disable	556
Table 3-835. Function usart_wakeup_mode_config	556
Table 3-836. Function usart_receive_fifo_enable.....	557
Table 3-837. Function usart_receive_fifo_disable.....	557
Table 3-838. Function usart_receive_fifo_counter_number	558
Table 3-839. Function usart_flag_get	559
Table 3-840. Function usart_flag_clear	559
Table 3-841. Function usart_interrupt_enable	560
Table 3-842. Function usart_interrupt_disable	561
Table 3-843. Function usart_interrupt_flag_get	561
Table 3-844. Function usart_interrupt_flag_clear	562
Table 3-845. WWDGT Registers	563
Table 3-846. WWDGT firmware function.....	564
Table 3-847. Function wwdgt_deinit	564
Table 3-848. Function wwdgt_enable	564
Table 3-849. Function wwdgt_counter_update.....	565
Table 3-850. Function wwdgt_config	565
Table 3-851. Function wwdgt_interrupt_enable	567
Table 3-852. Function wwdgt_flag_get.....	567
Table 3-853. Function wwdgt_flag_clear	568
Table 4-1. Revision history	569

1. Introduction

This manual introduces firmware library of GD32A513 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 GD32A513 devices are involved in the package. The peripheral driving code and firmware examples on evaluation board are also included in firmware library. Users need not learn each peripherals in details and it's easy to apply a peripheral by using the firmware library. Using firmware library can greatly reduce programming time, thereby reducing development costs.

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

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

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

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

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

1.1. Rules of User Manual and Firmware Library

1.1.1. Peripherals

Table 1-1. Peripherals

Peripherals	Descriptions
ADC	Analog-to-digital converter
BKP	Backup registers
CAN	Controller area network
CMP	Comparator
CRC	CRC calculation unit

Peripherals	Descriptions
DBG	Debug
DAC	Digital-to-analog converter
DMA	Direct memory access controller
DMAMUX	DMA request multiplexer
EXTI	Interrupt/event controller
FMC	Flash memory controller
FWDGT	Free watchdog timer
GPIO/AFIO	General-purpose and alternate-function I/Os
I2C	Inter-integrated circuit interface
MFCOM	Multi-function communication Interface
MISC	Nested Vectored Interrupt Controller
PMU	Power management unit
RCU	Reset and clock unit
RTC	Real-time Clock
SPI/I2S	Serial peripheral interface/Inter-IC sound
SYSCFG	System configuration
TIMER	TIMER
TRIGSEL	Trigger selection controller
USART	Universal synchronous/asynchronous receiver /transmitter
WWDGT	Window watchdog timer

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 “gd32a513_”, such as: gd32a513_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

GD32A513_Firmware_Library, the file structure is shown as below:

Figure 2-1. File structure of firmware library of GD32A513

- ▼  GD32E502_Firmware_Library
 - >  Docs
 - ▼  Examples
 - >  ADC
 - >  BKP
 - >  CAN
 - >  CMP
 - >  CRC
 - >  DAC
 - >  DBG
 - >  DMA
 - >  EXTI
 - >  FMC
 - >  FWDGT
 - >  GPIO
 - >  I2C
 - >  MFCOM
 - >  PMU
 - >  RCU
 - >  RTC
 - >  SPI
 - >  TIMER
 - >  TRIGSEL
 - >  USART
 - >  WWDGT
 - ▼  Firmware
 - >  CMSIS
 - >  GD32E502_standard_peripheral
 - ▼  Template
 -  IAR_project
 -  Keil_project
 -  Utilities

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;
- gd32a513_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);
- gd32a513_it.c: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- gd32a513_it.h: the header file include all the prototypes of the interrupt service routines;
- systick.c: the source file include the precise time delay functions by using systick;
- systick.h: the header file include the prototype of the precise time delay functions by using systick;
- main.c: example code. Note: all the examples are not influenced by software IDEs.

2.1.2. Firmware Folder

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

- CMSIS subfolder includes the Cortex M33 kernel support files, the startup file based on the Cortex M33 kernel processor, the global header file of GD32A513 and system configuration file;
- GD32A513_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;

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

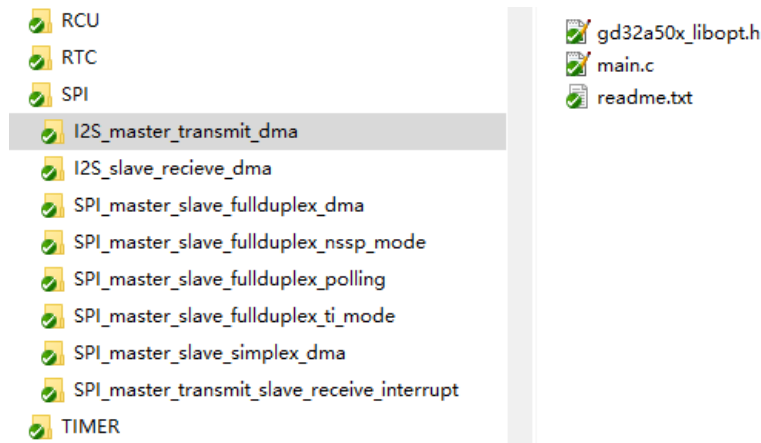
2.1.3. Template Folder

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 formware examples, the steps are shown as below:

Select files

Open “Examples” folder, select the module to be tested, such as SPI, open “SPI” folder, select an example of SPI, such as “I2S_master_transmit_dma”, 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 “I2S_master_transmit_dma” folder to the “Template” subfolder, shown as below:

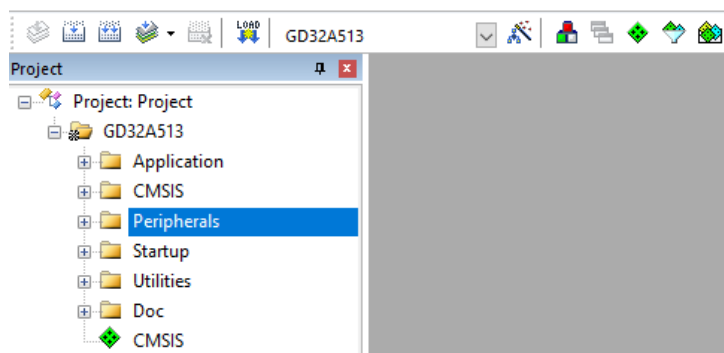
Figure 2-3. Copy the peripheral example files



Open a project

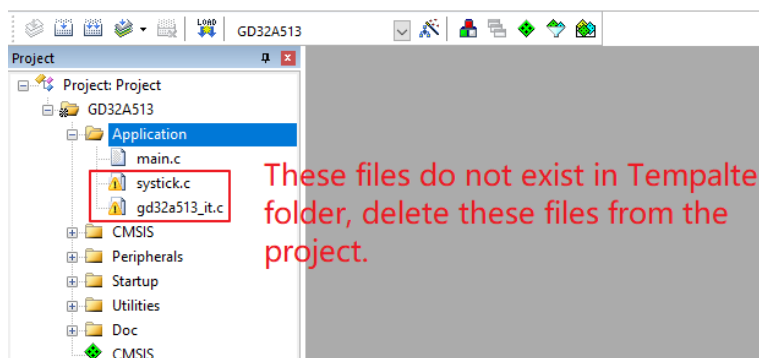
GD provides project in Keil and IAR, users can open project in different IDEs according to their need, such as ”Keil_project”, open \Template\Keil_project\Project.uvprojx, shown as below:

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:

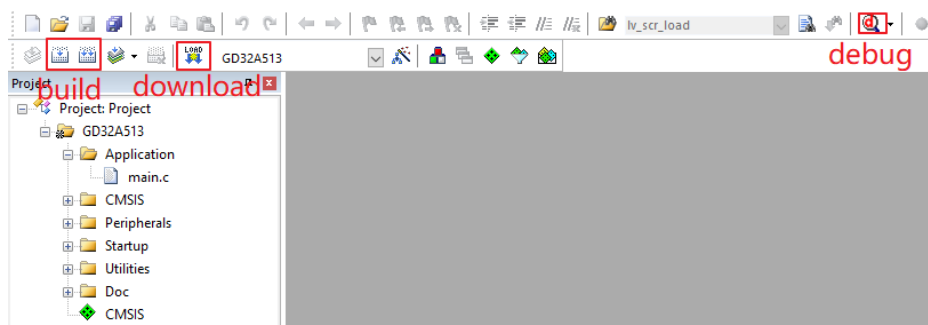
Figure 2-5. Configure project files



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. Utilities Folder

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

- gd32a513v_eval.h is related header file of the evaluation board about running the firmware examples;
- gd32a513v_eval.c is related source file 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
gd32a513_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.
gd32a513_it.h	Header file, including all the prototypes of interrupt service routines.
gd32a513_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.
gd32a513_xxx.h	The header file of peripheral xxx, including functions about peripheral xxx, and the variables used for functions.
gd32a513_xxx.c	The C source file for driving peripheral xxx.
systick.h	The header file of systick.c, including prototypes of systick configuration function and delay function.
systick.c	The source file about systick configuration function and delay function.
readme.txt	Description document about how to configure and how to use the firmware example.

3. Firmware Library of Standard Peripherals

3.1. Overview of Firmware Library of Standard Peripherals

The description format of firmware functions are shown as below:

Table 3-1. Peripheral function format of Firmware Library

Function name	Name of peripheral function
Function prototype	Declaration prototype
Function descriptions	Explain the function how to work
Precondition	Requirements should meet before calling this function
The called functions	Other firmware functions called in this 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)

Registers	Descriptions
ADC_WDHT0	Watchdog 0 high threshold register
ADC_WDLT0	Watchdog 0 low threshold register
ADC_RSQ0	Regular sequence register 0
ADC_RSQ1	Regular sequence register 1
ADC_RSQ2	Regular sequence register 2
ADC_ISQ	Inserted sequence register
ADC_IDATAx (x=0..3)	Inserted data register x(x=0..3)
ADC_RDATA	Regular data register
ADC_OVSAMPCTL	Oversample control register
ADC_WD1SR	ADC watchdog 1 channel selection register
ADC_WDT1	ADC watchdog threshold register 1

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_enable	enable ADC interface
adc_disable	disable ADC interface
adc_calibration_enable	ADC calibration and reset calibration
adc_dma_mode_enable	enable DMA request
adc_dma_mode_disable	disable DMA request
adc_tempsensor_enable	enable the temperature sensor channel
adc_tempsensor_disable	disable the temperature sensor channel
adc_vrefint_enable	enable the vrefint channel
adc_vrefint_disable	disable the vrefint channel
adc_discontinuous_mode_config	configure ADC discontinuous mode
adc_mode_config	configure the ADC sync mode
adc_special_function_config	enable or disable ADC special function
adc_data_alignment_config	configure ADCx 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_regular_data_read	read ADC regular group data register

Function name	Function description
adc_inserted_data_read	read ADC inserted group data register
adc_sync_mode_convert_value_read	read the last ADC0 and ADC1 conversion result data in sync mode
adc_watchdog0_single_channel_enable	configure ADC analog watchdog 0 single channel
adc_watchdog0_group_channel_enable	configure ADC analog watchdog 0 group channel
adc_watchdog0_disable	disable ADC analog watchdog 0
adc_watchdog1_channel_config	configure ADC analog watchdog 1 channel
adc_watchdog1_disable	disable ADC analog watchdog 1
adc_watchdog0_threshold_config	configure ADC analog watchdog 0 threshold
adc_watchdog1_threshold_config	configure ADC analog watchdog 1 threshold
adc_resolution_config	configure ADC resolution
adc_oversample_mode_config	configure ADC oversample mode
adc_oversample_mode_enable	enable ADC oversample mode
adc_oversample_mode_disable	disable ADC oversample mode
adc_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_deinit

The description of adc_deinit is shown as below:

Table 3-4. Function adc_deinit

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

Example:

```
/* reset ADC0 */
```

```
adc_deinit(ADC0);
```

adc_enable

The description of adc_enable is shown as below:

Table 3-5. Function adc_enable

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

Example:

```
/* enable ADC0 */
```

```
adc_enable(ADC0);
```

adc_disable

The description of adc_disable is shown as below:

Table 3-6. Function adc_disable

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

Example:


```
/* disable ADC0 */
```

```
adc_disable(ADC0);
```

adc_calibration_enable

The description of adc_calibration_enable is shown as below:

Table 3-7. Function adc_calibration_enable

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

Example:

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

```
adc_calibration_enable(ADC0);
```

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

Example:

```
/* enable ADC0 DMA request */
```

```
adc_dma_mode_enable(ADC0);
```

adc_dma_mode_disable

The description of adc_dma_mode_disable is shown as below:

Table 3-9. Function adc_dma_mode_disable

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

Example:

```
/* disable ADC0 DMA request */
```

```
adc_dma_mode_disable(ADC0);
```

adc_tempsensor_enable

The description of adc_tempsensor_enable is shown as below:

Table 3-10. Function adc_tempsensor_enable

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

Example:

```
/* enable the temperature sensor channel */
```

```
adc_tempsensor_enable();
```

adc_tempsensor_disable

The description of adc_tempsensor_disable is shown as below:

Table 3-11. Function adc_tempsensor_disable

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

Example:

```
/* disable the temperature sensor channel */
```

```
adc_tempsensor_disable();
```

adc_vrefint_enable

The description of adc_vrefint_enable is shown as below:

Table 3-12. Function adc_vrefint_enable

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

Example:

```
/* enable the vrefint channel */
```

```
adc_vrefint_enable();
```

adc_vrefint_disable

The description of adc_vrefint_disable is shown as below:

Table 3-13. Function adc_vrefint_disable

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

Example:

```
/* disable the vrefint channel */
adc_vrefint_disable();
```

adc_discontinuous_mode_config

The description of adc_discontinuous_mode_config is shown as below:

Table 3-14. Function adc_discontinuous_mode_config

Function name	adc_discontinuous_mode_config
Function prototype	void adc_discontinuous_mode_config(uint32_t adc_periph, uint8_t adc_channel_group, uint8_t length);
Function descriptions	configure ADC discontinuous mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
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 ADC0 regular channel group discontinuous mode */
```

```
adc_discontinuous_mode_config(ADC0, ADC_REGULAR_CHANNEL, 6);
```

adc_mode_config

The description of adc_mode_config is shown as below:

Table 3-15. Function adc_mode_config

Function name	adc_mode_config
Function prototype	void adc_mode_config(uint32_t mode);
Function descriptions	configure the ADCs sync mode
Precondition	-
The called functions	-
Input parameter{in}	
mode	ADC mode
ADC_MODE_FREE	all the ADCs work independently
ADC_DAUL_REGULAL _PARALLEL_INSERTE D_PARALLEL	ADC0 and ADC1 work in combined regular parallel + inserted parallel mode
ADC_DAUL_REGULAL _PARALLEL_INSERTE D_ROTATION	ADC0 and ADC1 work in combined regular parallel + trigger rotation mode
ADC_DAUL_INSERTE D_PARALLEL_REGUL AL_FOLLOWUP_FAST	ADC0 and ADC1 work in combined inserted parallel + follow-up fast mode
ADC_DAUL_INSERTE D_PARALLEL_REGUL AL_FOLLOWUP_SLO W	ADC0 and ADC1 work in combined inserted parallel + follow-up slow mode
ADC_DAUL_INSERTE D_PARALLEL	ADC0 and ADC1 work in inserted parallel mode only
ADC_DAUL_REGULAL _PARALLEL	ADC0 and ADC1 work in regular parallel mode only
ADC_DAUL_REGULAL _FOLLOWUP_FAST	ADC0 and ADC1 work in follow-up fast mode only

<i>ADC_DAUL_REGULAR_FOLLOWUP_SLOW</i>	ADC0 and ADC1 work in follow-up slow mode only
<i>ADC_DAUL_INSERTED_TRIGGERR_ROTATION</i>	ADC0 and ADC1 work in trigger rotation mode only
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
adc_mode_config(ADC_MODE_FREE);
```

adc_special_function_config

The description of `adc_special_function_config` is shown as below:

Table 3-16. Function `adc_special_function_config`

Function name	<code>adc_special_function_config</code>
Function prototype	<code>void adc_special_function_config(uint32_t adc_periph, uint32_t function, ControlStatus newvalue);</code>
Function descriptions	enable or disable ADC special function
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
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 ADC0 scan mode */
```

```
adc_special_function_config(ADC0, ADC_SCAN_MODE, ENABLE);
```

adc_data_alignment_config

The description of adc_data_alignment_config is shown as below:

Table 3-17. Function adc_data_alignment_config

Function name	adc_data_alignment_config
Function prototype	void adc_data_alignment_config(uint32_t adc_periph, uint32_t data_alignment);
Function descriptions	configure ADCx data alignment
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
data_alignment	data alignment select
ADC_DATAALIGN_RIGHT	right alignment
ADC_DATAALIGN_LEFT	left alignment
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 data alignment */
```

```
adc_data_alignment_config(ADC0, ADC_DATAALIGN_RIGHT);
```

adc_channel_length_config

The description of adc_channel_length_config is shown as below:

Table 3-18. Function adc_channel_length_config

Function name	adc_channel_length_config
Function prototype	void adc_channel_length_config(uint32_t adc_periph, uint8_t adc_channel_group, uint32_t length);
Function descriptions	configure the length of regular channel group or inserted channel group
Precondition	-

The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
length	the length of the channel, regular channel 1-16, inserted channel 1-4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the length of ADC0 regular channel */
```

```
adc_channel_length_config(ADC0, ADC_REGULAR_CHANNEL, 4);
```

adc_regular_channel_config

The description of adc_regular_channel_config is shown as below:

Table 3-19. Function adc_regular_channel_config

Function name	adc_regular_channel_config
Function prototype	void adc_regular_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	configure ADC regular channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
rank	the regular group sequence rank, this parameter must be between 0 to 15
Input parameter{in}	
adc_channel	the selected ADC channel
ADC_CHANNEL_x (x=0..17)	ADC Channelx (x=0..17) (x=16 and x=17 are only for ADC0)
Input parameter{in}	
sample_time	the sample time value

ADC_SAMPLETIME_2POINT5	2.5 cycles
ADC_SAMPLETIME_14POINT5	14.5 cycles
ADC_SAMPLETIME_27POINT5	27.5 cycles
ADC_SAMPLETIME_55POINT5	55.5 cycles
ADC_SAMPLETIME_83POINT5	83.5 cycles
ADC_SAMPLETIME_111POINT5	111.5 cycles
ADC_SAMPLETIME_143POINT5	143.5 cycles
ADC_SAMPLETIME_479POINT5	479.5 cycles
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 regular channel */
```

```
adc_regular_channel_config(ADC0, 1, ADC_CHANNEL_0, ADC_SAMPLETIME_2POINT5);
```

adc_inserted_channel_config

The description of adc_inserted_channel_config is shown as below:

Table 3-20. Function adc_inserted_channel_config

Function name	adc_inserted_channel_config
Function prototype	void adc_inserted_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
Function descriptions	configure ADC inserted channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
rank	the inserted group sequencer rank, this parameter must be between 0 to 3
Input parameter{in}	
adc_channel	the selected ADC channel

ADC_CHANNEL_x (x=0..17)	ADC Channelx (x=0..17)(x=16 and x=17 are only for ADC0)
Input parameter{in}	
sample_time	the sample time value
ADC_SAMPLETIME_2POINT5	2.5 cycles
ADC_SAMPLETIME_14POINT5	14.5 cycles
ADC_SAMPLETIME_27POINT5	27.5 cycles
ADC_SAMPLETIME_55POINT5	55.5 cycles
ADC_SAMPLETIME_83POINT5	83.5 cycles
ADC_SAMPLETIME_111POINT5	111.5 cycles
ADC_SAMPLETIME_143POINT5	143.5 cycles
ADC_SAMPLETIME_479POINT5	479.5 cycles
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 inserted channel */
```

```
adc_inserted_channel_config(ADC0, 1, ADC_CHANNEL_0, ADC_SAMPLETIME_2POINT5);
```

adc_inserted_channel_offset_config

The description of adc_inserted_channel_offset_config is shown as below:

Table 3-21. Function adc_inserted_channel_offset_config

Function name	adc_inserted_channel_offset_config
Function prototype	void adc_inserted_channel_offset_config(uint32_t adc_periph, uint8_t inserted_channel, uint16_t offset);
Function descriptions	configure ADC inserted channel offset
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection

Input parameter{in}	
inserted_channel	insert channel select
<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 ADC0 inserted channel offset */
```

```
adc_inserted_channel_offset_config(ADC0, ADC_INSERTED_CHANNEL_0, 100);
```

adc_external_trigger_config

The description of adc_external_trigger_config is shown as below:

Table 3-22. Function adc_external_trigger_config

Function name	adc_external_trigger_config
Function prototype	void adc_external_trigger_config(uint32_t adc_periph, uint8_t adc_channel_group, ControlStatus newvalue);
Function descriptions	configure ADC external trigger
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
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}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC0 inserted channel group external trigger */

adc_external_trigger_config(ADC0, ADC_INSERTED_CHANNEL_0, ENABLE);
```

adc_external_trigger_source_config

The description of adc_external_trigger_source_config is shown as below:

Table 3-23. Function adc_external_trigger_source_config

Function name	adc_external_trigger_source_config
Function prototype	void adc_external_trigger_source_config(uint32_t adc_periph, uint8_t adc_channel_group, uint32_t external_trigger_source);
Function descriptions	configure ADC external trigger source
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
adc_channel_group	select the channel group
ADC_REGULAR_CHANNEL	regular channel group
ADC_INSERTED_CHANNEL	inserted channel group
Input parameter{in}	
external_trigger_source	regular or inserted group trigger source
ADC0_1_EXTTRIG_REGULAR_TRIGSEL	TRIGSEL select for regular channel
ADC0_1_EXTTRIG_REGULAR_NONE	software trigger for regular channel
ADC0_1_EXTTRIG_INSERTED_TRIGSEL	TRIGSEL select for inserted channel
ADC0_1_EXTTRIG_INSERTED_NONE	software trigger for inserted channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 regular channel TRIGSEL select source */
```

```
adc_external_trigger_source_config(ADC0,ADC_REGULAR_CHANNEL,
ADC0_1_EXTTRIG_REGULAR_TRIGSEL);
```

adc_software_trigger_enable

The description of adc_software_trigger_enable is shown as below:

Table 3-24. Function adc_software_trigger_enable

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

Example:

```
/* enable ADC0 regular channel group software trigger */
adc_software_trigger_enable(ADC0, ADC_REGULAR_CHANNEL);
```

adc_regular_data_read

The description of adc_regular_data_read is shown as below:

Table 3-25. Function adc_regular_data_read

Function name	adc_regular_data_read
Function prototype	uint16_t adc_regular_data_read(uint32_t adc_periph);
Function descriptions	read ADC regular group data register
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral

<i>ADCx(x=0,1)</i>	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	ADC conversion value (0~0xFFFF)

Example:

```
/* read ADC0 regular group data register */

uint16_t adc_value = 0;

adc_value = adc_regular_data_read(ADC0);
```

adc_inserted_data_read

The description of `adc_inserted_data_read` is shown as below:

Table 3-26. Function `adc_inserted_data_read`

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

Example:

```
/* read ADC0 inserted group data register */

uint16_t adc_value = 0;

adc_value = adc_inserted_data_read (ADC0, ADC_INSERTED_CHANNEL_0);
```

adc_sync_mode_convert_value_read

The description of `adc_sync_mode_convert_value_read` is shown as below:

Table 3-27. Function `adc_sync_mode_convert_value_read`

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

Example:

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

`adc_watchdog0_single_channel_enable`

The description of `adc_watchdog0_single_channel_enable` is shown as below:

Table 3-28. Function `adc_watchdog0_single_channel_enable`

Function name	<code>adc_watchdog0_single_channel_enable</code>
Function prototype	<code>void adc_watchdog0_single_channel_enable(uint32_t adc_periph, uint8_t adc_channel);</code>
Function descriptions	configure ADC analog watchdog 0 single channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Input parameter{in}	
adc_channel	the selected ADC channel
<i>ADC_CHANNEL_x</i> (x=0..17)	ADC channelx(x=0..17) (x=16 and x=17 are only for ADC0)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 analog watchdog 0 single channel */
```

```
adc_watchdog0_single_channel_enable(ADC0, ADC_CHANNEL_1);
```

adc_watchdog0_group_channel_enable

The description of adc_watchdog0_group_channel_enable is shown as below:

Table 3-29. Function adc_watchdog0_group_channel_enable

Function name	adc_watchdog0_group_channel_enable
Function prototype	void adc_watchdog0_group_channel_enable(uint32_t adc_periph, uint8_t adc_channel_group);
Function descriptions	configure ADC analog watchdog 0 group channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
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 ADC0 analog watchdog 0 group channel */
```

```
adc_watchdog0_group_channel_enable(ADC0, ADC_REGULAR_CHANNEL);
```

adc_watchdog0_disable

The description of adc_watchdog0_disable is shown as below:

Table 3-30. Function adc_watchdog0_disable

Function name	adc_watchdog0_disable
Function prototype	void adc_watchdog0_disable(uint32_t adc_periph);
Function descriptions	disable ADC analog watchdog 0
Precondition	-
The called functions	-
Input parameter{in}	

adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 analog watchdog 0 */
```

```
adc_watchdog0_disable(ADC0);
```

adc_watchdog1_channel_config

The description of adc_watchdog1_channel_config is shown as below:

Table 3-31. Function adc_watchdog1_channel_config

Function name	adc_watchdog1_channel_config
Function prototype	void adc_watchdog1_channel_config(uint32_t adc_periph, uint32_t adc_channel, ControlStatus newvalue);
Function descriptions	configure ADC analog watchdog 1 channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Input parameter{in}	
adc_channel	the selected ADC channel
<i>ADC_AWD1_SELECT1 ON_CHANNEL_x (x=0..17), ADC_AWD1_SELECT1 ON_CHANNEL_ALL</i>	ADC channel analog watchdog 1 selection (x=0..17, x=16 and x=17 are only for ADC0)
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC analog watchdog 1 channel */
```

```
adc_watchdog1_channel_config(ADC0, ADC_AWD1_SELECTION_CHANNEL_1,
ENABLE);
```

adc_watchdog1_disable

The description of adc_watchdog1_disable is shown as below:

Table 3-32. Function adc_watchdog1_disable

Function name	adc_watchdog1_disable
Function prototype	void adc_watchdog1_disable(uint32_t adc_periph);
Function descriptions	disable ADC analog watchdog 1
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 analog watchdog 1 */
adc_watchdog1_disable(ADC0);
```

adc_watchdog0_threshold_config

The description of adc_watchdog0_threshold_config is shown as below:

Table 3-33. Function adc_watchdog0_threshold_config

Function name	adc_watchdog0_threshold_config
Function prototype	void adc_watchdog0_threshold_config(uint32_t adc_periph , uint16_t low_threshold , uint16_t high_threshold);
Function descriptions	configure ADC analog watchdog 0 threshold
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Input parameter{in}	
low_threshold	analog watchdog low threshold, 0..4095
Input parameter{in}	
high_threshold	analog watchdog high threshold, 0..4095
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure ADC0 analog watchdog 0 threshold */
```

```
adc_watchdog0_threshold_config(ADC0, 0x0400, 0x0A00);
```

adc_watchdog1_threshold_config

The description of adc_watchdog1_threshold_config is shown as below:

Table 3-34. Function adc_watchdog1_threshold_config

Function name	adc_watchdog1_threshold_config
Function prototype	void adc_watchdog1_threshold_config(uint32_t adc_periph , uint8_t low_threshold , uint8_t high_threshold);
Function descriptions	configure ADC analog watchdog 1 threshold
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Input parameter{in}	
low_threshold	analog watchdog low threshold, 0..255
Input parameter{in}	
high_threshold	analog watchdog high threshold, 0..255
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 analog watchdog 1 threshold */
```

```
adc_watchdog1_threshold_config(ADC0, 0x40, 0xA0);
```

adc_resolution_config

The description of adc_resolution_config is shown as below:

Table 3-35. Function adc_resolution_config

Function name	adc_resolution_config
Function prototype	void adc_resolution_config(uint32_t adc_periph , uint32_t resolution);
Function descriptions	configure ADC resolution

Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Input parameter{in}	
resolution	ADC resolution
<i>ADC_RESOLUTION_12B</i>	12-bit ADC resolution
<i>ADC_RESOLUTION_10B</i>	10-bit ADC resolution
<i>ADC_RESOLUTION_8B</i>	8-bit ADC resolution
<i>ADC_RESOLUTION_6B</i>	6-bit ADC resolution
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 resolution */
```

```
adc_resolution_config(ADC0, ADC_RESOLUTION_8B);
```

adc_oversample_mode_config

The description of adc_oversample_mode_config is shown as below:

Table 3-36. Function adc_oversample_mode_config

Function name	adc_oversample_mode_config
Function prototype	void adc_oversample_mode_config(uint32_t adc_periph, uint32_t mode, uint16_t shift, uint8_t ratio);
Function descriptions	configure ADC oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Input parameter{in}	
mode	ADC oversampling mode
<i>ADC_OVERSAMPLING_ALL_CONVERT</i>	all oversampled conversions for a channel are done consecutively after a trigger

ADC_OVERSAMPLING _ONE_CONVERT	each oversampled conversion for a channel needs a trigger
Input parameter{in}	
shift	ADC oversampling shift
ADC_OVERSAMPLING _SHIFT_NONE	no oversampling shift
ADC_OVERSAMPLING _SHIFT_1B	1-bit oversampling shift
ADC_OVERSAMPLING _SHIFT_2B	2-bit oversampling shift
ADC_OVERSAMPLING _SHIFT_3B	3-bit oversampling shift
ADC_OVERSAMPLING _SHIFT_4B	4-bit oversampling shift
ADC_OVERSAMPLING _SHIFT_5B	5-bit oversampling shift
ADC_OVERSAMPLING _SHIFT_6B	6-bit oversampling shift
ADC_OVERSAMPLING _SHIFT_7B	7-bit oversampling shift
ADC_OVERSAMPLING _SHIFT_8B	8-bit oversampling shift
Input parameter{in}	
ratio	ADC oversampling ratio
ADC_OVERSAMPLING _RATIO_MUL2	oversampling ratio multiple 2
ADC_OVERSAMPLING _RATIO_MUL4	oversampling ratio multiple 4
ADC_OVERSAMPLING _RATIO_MUL8	oversampling ratio multiple 8
ADC_OVERSAMPLING _RATIO_MUL16	oversampling ratio multiple 16
ADC_OVERSAMPLING _RATIO_MUL32	oversampling ratio multiple 32
ADC_OVERSAMPLING _RATIO_MUL64	oversampling ratio multiple 64
ADC_OVERSAMPLING _RATIO_MUL128	oversampling ratio multiple 128
ADC_OVERSAMPLING _RATIO_MUL256	oversampling ratio multiple 256
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* configure ADC1 oversample mode: 16 times sample, 4 bits shift */
```

```
adc_oversample_mode_config(ADC1, ADC_OVERSAMPLING_ALL_CONVERT,
ADC_OVERSAMPLING_SHIFT_4B, ADC_OVERSAMPLING_RATIO_MUL16);
```

adc_oversample_mode_enable

The description of adc_oversample_mode_enable is shown as below:

Table 3-37. Function adc_oversample_mode_enable

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

Example:

```
/* enable ADC0 oversample mode */
```

```
adc_oversample_mode_enable (ADC0);
```

adc_oversample_mode_disable

The description of adc_oversample_mode_disable is shown as below:

Table 3-38. Function adc_oversample_mode_disable

Function name	adc_oversample_mode_disable
Function prototype	void adc_oversample_mode_disable(uint32_t adc_periph);
Function descriptions	disable ADC oversample mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable ADC0 oversample mode */
```

```
adc_oversample_mode_disable (ADC0);
```

adc_flag_get

The description of adc_flag_get is shown as below:

Table 3-39. Function adc_flag_get

Function name	adc_flag_get
Function prototype	FlagStatus adc_flag_get(uint32_t adc_periph, uint32_t flag);
Function descriptions	get the ADC flag
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
Input parameter{in}	
flag	the adc flag bits
ADC_FLAG_WDE0	analog watchdog 0 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_WDE1	analog watchdog 1 event flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ADC0 analog watchdog 0 flag bits */
```

```
FlagStatus flag_value;
```

```
flag_value = adc_flag_get(ADC0, ADC_FLAG_WDE0);
```

adc_flag_clear

The description of adc_flag_clear is shown as below:

Table 3-40. Function `adc_flag_clear`

Function name	<code>adc_flag_clear</code>
Function prototype	<code>void adc_flag_clear(uint32_t adc_periph, uint32_t flag);</code>
Function descriptions	clear the ADC flag bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<code>ADCx(x=0,1)</code>	ADC peripheral selection
Input parameter{in}	
flag	the adc flag bits
<code>ADC_FLAG_WDE0</code>	analog watchdog 0 event flag
<code>ADC_FLAG_EOC</code>	end of group conversion flag
<code>ADC_FLAG_EOIC</code>	end of inserted group conversion flag
<code>ADC_FLAG_STIC</code>	start flag of inserted channel group
<code>ADC_FLAG_STRC</code>	start flag of regular channel group
<code>ADC_FLAG_WDE1</code>	analog watchdog 1 event flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the ADC0 analog watchdog 0 flag bits */
```

```
adc_flag_clear(ADC0, ADC_FLAG_WDE0);
```

adc_interrupt_enable

The description of `adc_interrupt_enable` is shown as below:

Table 3-41. Function `adc_interrupt_enable`

Function name	<code>adc_interrupt_enable</code>
Function prototype	<code>void adc_interrupt_enable(uint32_t adc_periph, uint32_t interrupt);</code>
Function descriptions	enable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<code>ADCx(x=0,1)</code>	ADC peripheral selection
Input parameter{in}	
interrupt	the adc interrupt
<code>ADC_INT_WDE0</code>	analog watchdog 0 interrupt
<code>ADC_INT_EOC</code>	end of group conversion interrupt

<i>ADC_INT_EOIC</i>	end of inserted group conversion interrupt
<i>ADC_INT_WDE1</i>	analog watchdog 1 interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC0 analog watchdog 0 interrupt */
adc_interrupt_enable(ADC0, ADC_INT_WDE0);
```

adc_interrupt_disable

The description of adc_interrupt_disable is shown as below:

Table 3-42. Function adc_interrupt_disable

Function name	adc_interrupt_disable
Function prototype	void adc_interrupt_disable(uint32_t adc_periph , uint32_t interrupt);
Function descriptions	disable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
Input parameter{in}	
interrupt	the adc interrupt
<i>ADC_INT_WDE0</i>	analog watchdog 0 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_WDE1</i>	analog watchdog 1 interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 analog watchdog 0 interrupt */
adc_interrupt_disable(ADC0, ADC_INT_WDE0);
```

adc_interrupt_flag_get

The description of adc_interrupt_flag_get is shown as below:

Table 3-43. Function `adc_interrupt_flag_get`

Function name	<code>adc_interrupt_flag_get</code>
Function prototype	<code>FlagStatus adc_interrupt_flag_get(uint32_t adc_periph, uint32_t int_flag);</code>
Function descriptions	get the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<code>ADCx(x=0,1)</code>	ADC peripheral selection
Input parameter{in}	
int_flag	the adc interrupt bits
<code>ADC_INT_FLAG_WDE0</code>	analog watchdog 0 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_WDE1</code>	analog watchdog 1 interrupt
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ADC0 analog watchdog 0 interrupt bits*/
```

```
FlagStatus flag_value;
```

```
flag_value = adc_interrupt_flag_get(ADC0, ADC_INT_WDE0);
```

adc_interrupt_flag_clear

The description of `adc_interrupt_flag_clear` is shown as below:

Table 3-44. Function `adc_interrupt_flag_clear`

Function name	<code>adc_interrupt_flag_clear</code>
Function prototype	<code>void adc_interrupt_flag_clear(uint32_t adc_periph, uint32_t int_flag);</code>
Function descriptions	clear the ADC interrupt bits
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
<code>ADCx(x=0,1)</code>	ADC peripheral selection
Input parameter{in}	
int_flag	the adc interrupt bits

ADC_INT_FLAG_WDE0	analog watchdog 0 interrupt
ADC_INT_FLAG_EOC	end of group conversion interrupt
ADC_INT_FLAG_EOIC	end of inserted group conversion interrupt
ADC_INT_FLAG_WDE1	analog watchdog 1 interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the ADC0 analog watchdog 0 interrupt bits*/
adc_interrupt_flag_clear(ADC0, ADC_INT_WDE0);
```

3.3. BKP

The Backup registers are located in the Backup domain that remains powered-on by VDD power, there are ten 16-bit (20 bytes) registers for data protection of user application data, and the wake-up action from standby mode or system reset do not affect these registers. The BKP registers are listed in chapter [3.3.1](#), the BKP firmware functions are introduced in chapter [3.3.2](#).

3.3.1. Descriptions of Peripheral registers

BKP registers are listed in the table shown as below:

Table 3-45. BKP Registers

Registers	Descriptions
BKP_DATAx (x= 0..9)	Backup data register
BKP_OCTL	RTC signal output control register
BKP_TPCTL	Tamper pin control register
BKP_TPCS	Tamper control and status register

3.3.2. Descriptions of Peripheral functions

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

Table 3-46. BKP firmware function

Function name	Function description
bkp_deinit	reset BKP registers
bkp_data_write	write BKP data register

Function name	Function description
bkp_data_read	read BKP data register
bkp_rtc_calibration_output_enable	enable RTC clock calibration output
bkp_rtc_calibration_output_disable	disable RTC clock calibration output
bkp_rtc_signal_output_enable	enable RTC alarm or second signal output
bkp_rtc_signal_output_disable	disable RTC alarm or second signal output
bkp_rtc_output_select	select RTC output, the RTC output can be select as alarm pulse or second pulse
bkp_rtc_clock_output_select	select RTC clock output
bkp_rtc_clock_calibration_direction	select RTC clock calibration direction
bkp_rtc_calibration_value_set	set RTC clock calibration value
bkp_osc32in_pin_select	select OSC32IN pin
bkp_tamper_detection_enable	enable tamper pin detection
bkp_tamper_detection_disable	disable tamper pin detection
bkp_tamper_active_level_set	set tamper pin active level
bkp_tamper_interrupt_enable	enable tamper interrupt
bkp_tamper_interrupt_disable	disable tamper interrupt
bkp_flag_get	get bkp flag state
bkp_flag_clear	clear bkp flag state
bkp_interrupt_flag_get	get bkp interrupt flag state
bkp_interrupt_flag_clear	clear bkp interrupt flag state

Enum bkp_data_register_enum

Table 3-47. Enum bkp_data_register_enum

Member name	Function description
BKP_DATA_0	bkp data register number 0
BKP_DATA_1	bkp data register number 1
BKP_DATA_2	bkp data register number 2
BKP_DATA_3	bkp data register number 3
BKP_DATA_4	bkp data register number 4
BKP_DATA_5	bkp data register number 5
BKP_DATA_6	bkp data register number 6
BKP_DATA_7	bkp data register number 7
BKP_DATA_8	bkp data register number 8
BKP_DATA_9	bkp data register number 9

bkp_deinit

The description of bkp_deinit is shown as below:

Table 3-48. Function bkp_deinit

Function name	bkp_deinit
Function prototype	void bkp_deinit(void);

Function descriptions	reset BKP registers
Precondition	-
The called functions	rcu_bkp_reset_enable / rcu_bkp_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset BKP registers */
```

```
bkp_deinit ();
```

bkp_data_write

The description of bkp_data_write is shown as below:

Table 3-49. Function bkp_data_write

Function name	bkp_data_write
Function prototype	void bkp_data_write(bkp_data_register_enum register_number, uint16_t data);
Function descriptions	write BKP data register
Precondition	-
The called functions	-
Input parameter{in}	
register_number	refer to Table 3-47. Enum bkp_data_register_enum
<i>BKP_DATA_x(x = 0..9)</i>	bkp data register number x
Input parameter{in}	
data	the data to be write in BKP data register
<i>0-0xffff</i>	data value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write BKP data register */
```

```
bkp_write_data (BKP_DATA_0, 0x1226);
```

bkp_data_read

The description of bkp_data_read is shown as below:

Table 3-50. Function bkp_data_read

Function name	bkp_data_read
Function prototype	uint16_t bkp_data_read(bkp_data_register_enum register_number);
Function descriptions	read BKP data register
Precondition	-
The called functions	-
Input parameter{in}	
register_number	refer to Table 3-47. Enum bkp_data_register_enum
BKP_DATA_x(x = 0..9)	bkp data register number x
Output parameter{out}	
-	-
Return value	
uint16_t	0-0xffff

Example:

```
/* read BKP data register */
uint16_t data;
data = bkp_read_data (BKP_DATA_0);
```

bkp_rtc_calibration_output_enable

The description of bkp_rtc_calibration_output_enable is shown as below:

Table 3-51. Function bkp_rtc_calibration_output_enable

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

Example:

```
/* enable RTC clock calibration output */
bkp_rtc_calibration_output_enable();
```

bkp_rtc_calibration_output_disable

The description of bkp_rtc_calibration_output_disable is shown as below:

Table 3-52. Function bkp_rtc_calibration_output_disable

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

Example:

```
/* disable RTC clock calibration output */
bkp_rtc_calibration_output_disable();
```

bkp_rtc_signal_output_enable

The description of bkp_rtc_signal_output_enable is shown as below:

Table 3-53. Function bkp_rtc_signal_output_enable

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

Example:

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

bkp_rtc_signal_output_disable

The description of bkp_rtc_signal_output_disable is shown as below:

Table 3-54. Function bkp_rtc_signal_output_disable

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

Example:

```
/* disable RTC alarm or second signal output */
```

```
bkp_rtc_signal_output_disable();
```

bkp_rtc_output_select

The description of bkp_rtc_output_select is shown as below:

Table 3-55. Function bkp_rtc_output_select

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

Example:

```
/* select RTC output alarm signal output */
```

```
bkp_rtc_output_select (RTC_OUTPUT_ALARM_PULSE);
```


bkp_rtc_clock_output_select

The description of bkp_rtc_clock_output_select is shown as below:

Table 3-56. Function bkp_rtc_clock_output_select

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

Example:

```
/* select RTC clock divided 64 to output */
```

```
bkp_rtc_clock_output_select (RTC_CLOCK_DIV_64);
```

bkp_rtc_clock_calibration_direction

The description of bkp_rtc_clock_calibration_direction is shown as below:

Table 3-57. Function bkp_rtc_clock_calibration_direction

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

-	-
---	---

Example:

```
/* set RTC clock slowed down */
```

```
bkp_rtc_clock_calibration_direction (RTC_CLOCK_SLOWED_DOWN);
```

bkp_rtc_calibration_value_set

The description of bkp_rtc_calibration_value_set is shown as below:

Table 3-58. Function bkp_rtc_calibration_value_set

Function name	bkp_rtc_calibration_value_set
Function prototype	void bkp_rtc_calibration_value_set(uint8_t value);
Function descriptions	set RTC clock calibration value
Precondition	-
The called functions	-
Input parameter{in}	
value	RTC clock calibration value
0x00 - 0x7F	value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set RTC clock calibration value */
```

```
bkp_rtc_calibration_value_set (0x7f);
```

bkp_osc32in_pin_select

The description of bkp_osc32in_pin_select is shown as below:

Table 3-59. bkp_osc32in_pin_select

Function name	bkp_osc32in_pin_select
Function prototype	void bkp_osc32in_pin_select(uint16_t inputpin);
Function descriptions	select OSC32IN pin
Precondition	-
The called functions	-
Input parameter{in}	
inputpin	select OSC32IN pin
OSC32IN_PC13	OSC32IN pin is PC13
OSC32IN_PC14	OSC32IN pin is PC14
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* select OSC32IN pin */
```

```
bkp_osc32in_pin_select (OSC32IN pin is PC14);
```

bkp_tamper_detection_enable

The description of bkp_tamper_detection_enable is shown as below:

Table 3-60. Function bkp_tamper_detection_enable

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

Example:

```
/* enable tamper pin detection */
```

```
bkp_tamper_detection_enable();
```

bkp_tamper_detection_disable

The description of bkp_tamper_detection_disable is shown as below:

Table 3-61. Function bkp_tamper_detection_disable

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

Return value	
-	-

Example:

```
/* disable tamper pin detection */
```

```
bkp_tamper_detection_disable();
```

bkp_tamper_active_level_set

The description of bkp_tamper_active_level_set is shown as below:

Table 3-62. Function bkp_tamper_active_level_set

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

Example:

```
/* set tamper pin active level high */
```

```
bkp_tamper_active_level_set (TAMPER_PIN_ACTIVE_HIGH);
```

bkp_tamper_interrupt_enable

The description of bkp_tamper_interrupt_enable is shown as below:

Table 3-63. Function bkp_tamper_interrupt_enable

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

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

Example:

```
/* enable tamper pin interrupt */
```

```
bkp_tamper_interrupt_enable ();
```

bkp_tamper_interrupt_disable

The description of bkp_tamper_interrupt_disable is shown as below:

Table 3-64. Function bkp_tamper_interrupt_disable

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

Example:

```
/* disable tamper pin interrupt */
```

```
bkp_tamper_interrupt_disable ();
```

bkp_flag_get

The description of bkp_flag_get is shown as below:

Table 3-65. Function bkp_flag_get

Function name	bkp_flag_get
Function prototype	FlagStatus bkp_flag_get(uint16_t flag);
Function descriptions	get bkp flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	bkp flag state

<i>BKP_FLAG_TAMPER</i>	tamper event flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get BKP flag state */
```

```
FlagStatus status;
```

```
status = bkp_flag_get (BKP_FLAG_TAMPER);
```

bkp_flag_clear

The description of bkp_flag_clear is shown as below:

Table 3-66. Function bkp_flag_clear

Function name	bkp_flag_clear
Function prototype	void bkp_flag_clear(uint16_t flag);
Function descriptions	clear bkp flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	bkp flag state
<i>BKP_FLAG_TAMPER</i>	tamper event flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear BKP flag state */
```

```
bkp_flag_clear (BKP_FLAG_TAMPER);
```

bkp_interrupt_flag_get

The description of bkp_interrupt_flag_get is shown as below:

Table 3-67. Function bkp_interrupt_flag_get

Function name	bkp_interrupt_flag_get
Function prototype	FlagStatus bkp_interrupt_flag_get(uint16_t flag);
Function descriptions	get bkp interrupt flag state
Precondition	-
The called functions	-

Input parameter{in}	
flag	bkp interrupt flag state
<i>BKP_INT_FLAG_TAMP ER</i>	tamper interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get BKP interrupt flag state */
```

```
bkp_interrupt_flag_get (BKP_INT_FLAG_TAMPER);
```

bkp_interrupt_flag_clear

The description of bkp_interrupt_flag_clear is shown as below:

Table 3-68. Function bkp_interrupt_flag_clear

Function name	bkp_interrupt_flag_clear
Function prototype	void bkp_interrupt_flag_clear(uint16_t flag);
Function descriptions	clear bkp interrupt flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	bkp interrupt flag state
<i>BKP_INT_FLAG_TAMP ER</i>	tamper interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear BKP interrupt flag state */
```

```
bkp_interrupt_flag_clear (BKP_INT_FLAG_TAMPER);
```

3.4. CAN

CAN bus (Controller Area Network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer. The CAN interface supports the CAN 2.0A/B protocol, ISO 11898-1:2015 and BOSCH CAN FD specification. The CAN registers are listed in chapter [3.4.1](#), the CAN firmware functions are introduced in chapter

[3.4.2.](#)

3.4.1. Descriptions of Peripheral registers

CAN registers are listed in the table shown as below:

Table 3-69. CAN Registers

Registers	Descriptions
CAN_CTL0	CAN control register 0
CAN_CTL1	CAN control register 1
CAN_TIMER	CAN timer register
CAN_RMPUBF	CAN receive mailbox public filter register
CAN_ERR0	CAN error register 0
CAN_ERR1	CAN error register 1
CAN_INTEN	CAN interrupt enable register
CAN_STAT	CAN status register
CAN_CTL2	CAN control register 2
CAN_CRCC	CAN crc for classical frame register
CAN_RFIFOPUBF	CAN receive fifo public filter register
CAN_RFIFOIFMN	CAN receive fifo identifier filter matching number register
CAN_BT	CAN bit timing register
CAN_RFIFOMPFX (x = 0..31)	CAN receive fifo / mailbox private filter x register
CAN_PN_CTL0	Pretended Networking mode control register 0
CAN_PN_TO	Pretended Networking mode timeout register
CAN_PN_STAT	Pretended Networking mode status register
CAN_PN_EID0	Pretended Networking mode expected identifier 0 register
CAN_PN_EDLC	Pretended Networking mode expected dlc register
CAN_PN_EDL0	Pretended Networking mode expected data low 0 register
CAN_PN_EDL1	Pretended Networking mode expected data low 1 register
CAN_PN_IFEID1	Pretended Networking mode identifier filter / expected identifier 1 register
CAN_PN_DF0EDH 0	Pretended Networking mode data 0 filter / expected data high 0 register
CAN_PN_DF1EDH 1	Pretended Networking mode data 1 filter / expected data high 1 register
CAN_PN_RWMxCS (x = 0..3)	Pretended Networking mode received wakeup mailbox x control status information register
CAN_PN_RWMxI (x = 0..3)	Pretended Networking mode received wakeup mailbox x identifier register
CAN_PN_RWMxD0 (x = 0..3)	Pretended Networking mode received wakeup mailbox x data 0 register
CAN_PN_RWMxD1 (x = 0..3)	Pretended Networking mode received wakeup mailbox x data 1 register

Registers	Descriptions
CAN_FDCTL	CAN FD control register
CAN_FDBT	CAN bit timing register
CAN_CRCCFD	CAN CRC for classical and FD frame register

3.4.2. Descriptions of Peripheral functions

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

Table 3-70. CAN firmware function

Function name	Function description
can_deinit	deinitialize CAN
can_software_reset	reset CAN internal state machines and CAN registers
can_init	CAN module initialization
can_struct_para_init	initialize CAN parameter structure with a default value
can_private_filter_config	configure receive fifo/mailbox private filter
can_operation_mode_enter	enter the corresponding mode
can_operation_mode_get	get operation mode
can_inactive_mode_exit	exit inactive mode
can_pn_mode_exit	exit Pretended Networking mode
can_fd_config	can FD initialize
can_bitrate_switch_enable	enable bit rate switching
can_bitrate_switch_disable	disable bit rate switching
can_tdc_get	get transmitter delay compensation value
can_tdc_enable	enable transmitter delay compensation
can_tdc_disable	disable transmitter delay compensation
can_rx_fifo_config	configure rx FIFO
can_rx_fifo_filter_table_config	configure rx FIFO filter table
can_rx_fifo_read	read rx FIFO data
can_rx_fifo_filter_matching_number_get	get rx FIFO filter matching number
can_rx_fifo_clear	clear rx FIFO
can_ram_address_get	get mailbox RAM address
can_mailbox_config	config mailbox
can_mailbox_transmit_abort	abort mailbox transmit
can_mailbox_transmit_inactive	inactive transmit mailbox
can_mailbox_receive_data_read	read receive mailbox data
can_mailbox_receive_lock	lock the receive mailbox
can_mailbox_receive_unlock	unlock the receive mailbox
can_mailbox_receive_inactive	inactive the receive mailbox
can_mailbox_code_get	get mailbox code value
can_error_counter_config	configure error counter
can_error_counter_get	get error count

Function name	Function description
can_error_state_get	get error state indicator
can_crc_get	get mailbox CRC value
can_pn_mode_config	configure Pretended Networking mode parameter
can_pn_mode_filter_config	configure pn mode filter
can_pn_mode_num_of_match_get	get matching message counter of Pretended Networking mode
can_pn_mode_data_read	get matching message
can_self_reception_enable	enable self reception
can_self_reception_disable	disable self reception
can_transmit_abort_enable	enable transmit abort
can_transmit_abort_disable	disable transmit abort
can_auto_busoff_recovery_enable	enable auto bus off recovery mode
can_auto_busoff_recovery_disable	disable auto bus off recovery mode
can_time_sync_enable	enable time sync mode
can_time_sync_disable	disable time sync mode
can_edge_filter_mode_enable	enable edge filter mode
can_edge_filter_mode_disable	disable edge filter mode
can_ped_mode_enable	enable protocol exception detection mode
can_ped_mode_disable	disable protocol exception detection mode
can_arbitration_delay_bits_config	configure arbitration delay bits
can_bsp_mode_config	configure bit sampling mode
can_flag_get	get CAN flag
can_flag_clear	clear CAN flag
can_interrupt_enable	enable CAN interrupt
can_interrupt_disable	disable CAN interrupt
can_interrupt_flag_get	get CAN interrupt flag
can_interrupt_flag_clear	clear CAN interrupt flag

Structure can_error_counter_struct

Table 3-71. Structure can_error_counter_struct

Member name	Function description
fd_data_phase_rx_errcnt	receive error counter for data phase of FD frames with BRS bit set
fd_data_phase_tx_errcnt	transmit error count for the data phase of FD frames with BRS bit set
rx_errcnt	receive error count defined by the CAN standard
tx_errcnt	transmit error count defined by the CAN standard

Structure can_parameter_struct

Table 3-72. Structure can_parameter_struct

Member name	Function description
internal_counter_source	internal counter source
mb_tx_order	mailbox transmit order
mb_rx_ide_rtr_type	IDE and RTR field filter type
mb_remote_frame	remote request frame is stored
self_reception	enable or disable self reception
mb_tx_abort_enable	enable or disable transmit abort
local_priority_enable	enable or disable local priority
rx_private_filter_queue_enable	private filter and queue enable
edge_filter_enable	edge filter enable
protocol_exception_enable	protocol exception enable
rx_filter_order	receive filter order
memory_size	memory size
mb_public_filter	mailbox public filter
prescaler	baudrate prescaler
resync_jump_width	resynchronization jump width
prop_time_segment	propagation time segment
time_segment_1	time segment 1
time_segment_2	time segment 2

Structure can_mailbox_descriptor_struct

Table 3-73. Structure can_mailbox_descriptor_struct

Member name	Function description
timestamp	free-running counter timestamp
dlc	data length code in bytes
rtr	remote transmission request
ide	ID extended bit
srr	substitute remote request
code	mailbox code
esi	error state indicator
brs	bit rate switch
fdf	FD format indicator
id	identifier for frame
prio	local priority
data	data
data_bytes	data bytes

Member name	Function description
padding	FD mode padding data

Structure can_rx_fifo_struct

Table 3-74. Structure can_rx_fifo_struct

Member name	Function description
timestamp	free-running counter timestamp
dlc	data length code in bytes
rtr	remote transmission request
ide	ID extended bit
srr	substitute remote request
idhit	identifier filter matching number
id	identifier for frame
data[2]	fifo data

Structure can_fd_parameter_struct

Table 3-75. Structure can_fd_parameter_struct

Member name	Function description
iso_can_fd_enable	ISO CAN FD protocol enable
bitrate_switch_enable	data bit rate switch
mailbox_data_size	mailbox data size
tdc_enable	transmitter delay compensation enable
tdc_offset	transmitter delay compensation offset
prescaler	baudrate prescaler
resync_jump_width	resynchronization jump width
prop_time_segment	propagation time segment
time_segment_1	time segment 1
time_segment_2	time segment 2

Structure can_rx_fifo_id_filter_struct

Table 3-76. Structure can_rx_fifo_id_filter_struct

Member name	Function description
remote_frame	expected remote frame
extended_frame	expected extended frame
id	expected id

Structure can_fifo_parameter_struct

Table 3-77. Structure can_fifo_parameter_struct

Member name	Function description
dma_enable	DMA enable
filter_format_and_number	FIFO ID filter format and number
fifo_public_filter	FIFO ID public filter

Structure can_pn_mode_filter_struct

Table 3-78. Structure can_pn_mode_filter_struct

Member name	Function description
rtr	remote frame
ide	extended frame
id	id
dlc_high_threshold	DLC expected high threshold
dlc_low_threshold	DLC expected low threshold
payload[2]	data

Structure can_pn_mode_config_struct

Table 3-79. Structure can_pn_mode_config_struct

Member name	Function description
timeout_int	enable or disable timeout interrupt
match_int	enable or disable match interrupt
num_matches	set number of message matching times
match_timeout	set wakeup timeout value
frame_filter	set frame filtering type
id_filter	set id filtering type
data_filter	set data filtering type

Structure can_crc_struct

Table 3-80. Structure can_crc_struct

Member name	Function description
classical_frm_mb_number	associated number of mailbox for transmitting the CRCTC[14:0] value
classical_frm_transmitted_crc	transmitted CRC value for classical frames
classical_fd_frm_mb_number	associated number of mailbox for transmitting the CRCTCI[20:0] value
classical_fd_frm_transmitted_crc	transmitted CRC value for classical and ISO / non-ISO FD frames

Enum can_interrupt_enum

Table 3-81. Enum can_interrupt_enum

Member name	Function description
CAN_INT_RX_WARNING	receive warning interrupt
CAN_INT_TX_WARNING	transmit warning interrupt
CAN_INT_ERR_SUMMARY	error interrupt
CAN_INT_BUSOFF	bus off interrupt
CAN_INT_BUSOFF_RECOVERY	bus off recovery interrupt
CAN_INT_ERR_SUMMARY_FD	fd error interrupt
CAN_INT_MB0	mailbox 0 interrupt
CAN_INT_MB1	mailbox 1 interrupt
CAN_INT_MB2	mailbox 2 interrupt
CAN_INT_MB3	mailbox 3 interrupt
CAN_INT_MB4	mailbox 4 interrupt
CAN_INT_MB5	mailbox 5 interrupt
CAN_INT_MB6	mailbox 6 interrupt
CAN_INT_MB7	mailbox 7 interrupt
CAN_INT_MB8	mailbox 8 interrupt
CAN_INT_MB9	mailbox 9 interrupt
CAN_INT_MB10	mailbox 10 interrupt
CAN_INT_MB11	mailbox 11 interrupt
CAN_INT_MB12	mailbox 12 interrupt
CAN_INT_MB13	mailbox 13 interrupt
CAN_INT_MB14	mailbox 14 interrupt
CAN_INT_MB15	mailbox 15 interrupt
CAN_INT_MB16	mailbox 16 interrupt
CAN_INT_MB17	mailbox 17 interrupt
CAN_INT_MB18	mailbox 18 interrupt
CAN_INT_MB19	mailbox 19 interrupt
CAN_INT_MB20	mailbox 20 interrupt
CAN_INT_MB21	mailbox 21 interrupt
CAN_INT_MB22	mailbox 22 interrupt
CAN_INT_MB23	mailbox 23 interrupt
CAN_INT_MB24	mailbox 24 interrupt
CAN_INT_MB25	mailbox 25 interrupt
CAN_INT_MB26	mailbox 26 interrupt
CAN_INT_MB27	mailbox 27 interrupt

Member name	Function description
CAN_INT_MB28	mailbox 28 interrupt
CAN_INT_MB29	mailbox 29 interrupt
CAN_INT_MB30	mailbox 30 interrupt
CAN_INT_MB31	mailbox 31 interrupt
CAN_INT_FIFO_AVAILABLE	fifo available interrupt
CAN_INT_FIFO_WARNING	fifo warning interrupt
CAN_INT_FIFO_OVERFLOW	fifo overflow interrupt
CAN_INT_WAKEUP_MATCH	Pretended Networking match interrupt
CAN_INT_WAKEUP_TIMEOUT	Pretended Networking timeout wakeup interrupt

Enum can_flag_enum

Table 3-82. Enum can_flag_enum

Member name	Function description
CAN_FLAG_CAN_PN	Pretended Networking state flag
CAN_FLAG_SOFT_RST	software reset flag
CAN_FLAG_ERR_SUMMARY	error summary flag
CAN_FLAG_BUSOFF	bus off flag
CAN_FLAG_RECEIVING	receiving state flag
CAN_FLAG_TRANSMITTING	transmitting state flag
CAN_FLAG_IDLE	IDLE state flag
CAN_FLAG_RX_WARNING	receive warning flag
CAN_FLAG_TX_WARNING	transmit warning flag
CAN_FLAG_STUFF_ERR	stuff error flag
CAN_FLAG_FORM_ERR	form error flag
CAN_FLAG_CRC_ERR	CRC error flag

Member name	Function description
CAN_FLAG_ACK_ERR	ACK error flag
CAN_FLAG_BIT_DOMINANT_ERR	bit dominant error flag
CAN_FLAG_BIT_RECESSIVE_ERR	bit recessive error flag
CAN_FLAG_SYNC_ERR	synchronization flag
CAN_FLAG_BUSOFF_RECOVERY	bus off recovery flag
CAN_FLAG_ERR_SUMMARY_FD	FD error summary flag
CAN_FLAG_ERR_OVERRUN	error overrun flag
CAN_FLAG_STUFF_ERR_FD	stuff error in FD data phase flag
CAN_FLAG_FORM_ERR_FD	form error in FD data phase flag
CAN_FLAG_CRC_ERR_FD	CRC error in FD data phase flag
CAN_FLAG_BIT_DOMINANT_ERR_FD	bit dominant error in FD data phase flag
CAN_FLAG_BIT_RECESSIVE_ERR_FD	bit recessive error in FD data phase flag
CAN_FLAG_MB0	mailbox 0 flag
CAN_FLAG_MB1	mailbox 1 flag
CAN_FLAG_MB2	mailbox 2 flag
CAN_FLAG_MB3	mailbox 3 flag
CAN_FLAG_MB4	mailbox 4 flag
CAN_FLAG_MB5	mailbox 5 flag
CAN_FLAG_MB6	mailbox 6 flag
CAN_FLAG_MB7	mailbox 7 flag
CAN_FLAG_MB8	mailbox 8 flag
CAN_FLAG_MB9	mailbox 9 flag
CAN_FLAG_MB10	mailbox 10 flag
CAN_FLAG_MB11	mailbox 11 flag
CAN_FLAG_MB12	mailbox 12 flag
CAN_FLAG_MB13	mailbox 13 flag
CAN_FLAG_MB14	mailbox 14 flag
CAN_FLAG_MB15	mailbox 15 flag
CAN_FLAG_MB16	mailbox 16 flag

Member name	Function description
CAN_FLAG_MB17	mailbox 17 flag
CAN_FLAG_MB18	mailbox 18 flag
CAN_FLAG_MB19	mailbox 19 flag
CAN_FLAG_MB20	mailbox 20 flag
CAN_FLAG_MB21	mailbox 21 flag
CAN_FLAG_MB22	mailbox 22 flag
CAN_FLAG_MB23	mailbox 23 flag
CAN_FLAG_MB24	mailbox 24 flag
CAN_FLAG_MB25	mailbox 25 flag
CAN_FLAG_MB26	mailbox 26 flag
CAN_FLAG_MB27	mailbox 27 flag
CAN_FLAG_MB28	mailbox 28 flag
CAN_FLAG_MB29	mailbox 29 flag
CAN_FLAG_MB30	mailbox 30 flag
CAN_FLAG_MB31	mailbox 31 flag
CAN_FLAG_FIFO_AVAILABLE	fifo available flag
CAN_FLAG_FIFO_WARNING	fifo warning flag
CAN_FLAG_FIFO_OVERFLOW	fifo overflow flag
CAN_FLAG_WAKEUP_MATCH	Pretended Networking match flag
CAN_FLAG_WAKEUP_TIMEOUT	Pretended Networking timeout wakeup flag
CAN_FLAG_TDC_OUT_OF_RANGE	transmitter delay is out of compensation range flag

Enum can_interrupt_flag_enum

Table 3-83. Enum can_interrupt_flag_enum

Member name	Function description
CAN_INT_FLAG_ERROR_SUMMARY	error summary interrupt flag
CAN_INT_FLAG_BUS_OFF	bus off interrupt flag
CAN_INT_FLAG_RX_WARNING	receive warning interrupt flag
CAN_INT_FLAG_TX_WARNING	transmit warning interrupt flag
CAN_INT_FLAG_BUS_OFF_RECOVER	bus off recovery interrupt flag

Member name	Function description
Y	
CAN_INT_FLAG_ERR_SUMMARY_FD	fd error summary interrupt flag
CAN_INT_FLAG_MB0	mailbox 0 interrupt flag
CAN_INT_FLAG_MB1	mailbox 1 interrupt flag
CAN_INT_FLAG_MB2	mailbox 2 interrupt flag
CAN_INT_FLAG_MB3	mailbox 3 interrupt flag
CAN_INT_FLAG_MB4	mailbox 4 interrupt flag
CAN_INT_FLAG_MB5	mailbox 5 interrupt flag
CAN_INT_FLAG_MB6	mailbox 6 interrupt flag
CAN_INT_FLAG_MB7	mailbox 7 interrupt flag
CAN_INT_FLAG_MB8	mailbox 8 interrupt flag
CAN_INT_FLAG_MB9	mailbox 9 interrupt flag
CAN_INT_FLAG_MB10	mailbox 10 interrupt flag
CAN_INT_FLAG_MB11	mailbox 11 interrupt flag
CAN_INT_FLAG_MB12	mailbox 12 interrupt flag
CAN_INT_FLAG_MB13	mailbox 13 interrupt flag
CAN_INT_FLAG_MB14	mailbox 14 interrupt flag
CAN_INT_FLAG_MB15	mailbox 15 interrupt flag
CAN_INT_FLAG_MB16	mailbox 16 interrupt flag
CAN_INT_FLAG_MB17	mailbox 17 interrupt flag
CAN_INT_FLAG_MB18	mailbox 18 interrupt flag
CAN_INT_FLAG_MB19	mailbox 19 interrupt flag

Member name	Function description
B19	
CAN_INT_FLAG_M B20	mailbox 20 interrupt flag
CAN_INT_FLAG_M B21	mailbox 21 interrupt flag
CAN_INT_FLAG_M B22	mailbox 22 interrupt flag
CAN_INT_FLAG_M B23	mailbox 23 interrupt flag
CAN_INT_FLAG_M B24	mailbox 24 interrupt flag
CAN_INT_FLAG_M B25	mailbox 25 interrupt flag
CAN_INT_FLAG_M B26	mailbox 26 interrupt flag
CAN_INT_FLAG_M B27	mailbox 27 interrupt flag
CAN_INT_FLAG_M B28	mailbox 28 interrupt flag
CAN_INT_FLAG_M B29	mailbox 29 interrupt flag
CAN_INT_FLAG_M B30	mailbox 30 interrupt flag
CAN_INT_FLAG_M B31	mailbox 31 interrupt flag
CAN_INT_FLAG_FI FO_AVAILABLE	fifo available interrupt flag
CAN_INT_FLAG_FI FO_WARNING	fifo warning interrupt flag
CAN_INT_FLAG_FI FO_OVERFLOW	fifo overflow interrupt flag
CAN_INT_FLAG_W AKEUP_MATCH	Pretended Networking match interrupt flag
CAN_INT_FLAG_W AKEUP_TIMEOUT	Pretended Networking timeout wakeup interrupt flag

Enum can_operation_modes_enum

Table 3-84. Enum can_operation_modes_enum

Member name	Function description
CAN_NORMAL_MO DE	normal mode

Member name	Function description
CAN_MONITOR_MODE	monitor mode
CAN_LOOPBACK_SILENT_MODE	loopback mode
CAN_INACTIVE_MODE	inactive mode
CAN_DISABLE_MODE	disable mode
CAN_PN_MODE	Pretended Networking mode

Enum can_struct_type_enum

Table 3-85. Enum can_struct_type_enum

Member name	Function description
CAN_INIT_STRUCT	CAN initialize parameters struct
CAN_FD_INIT_STRUCT	CAN FD parameters struct
CAN_FIFO_INIT_STRUCT	CAN fifo parameters struct
CAN_PN_MODE_INIT_STRUCT	Pretended Networking mode parameter struct
CAN_PN_MODE_FILTER_STRUCT	Pretended Networking mode filter parameter struct

Enum can_error_state_enum

Table 3-86. Enum can_error_state_enum

Member name	Function description
CAN_ERROR_STATE_ACTIVE	CAN in error active
CAN_ERROR_STATE_PASSIVE	CAN in error passive
CAN_ERROR_STATE_BUS_OFF	CAN in bus off

can_deinit

The description of can_deinit is shown as below:

Table 3-87. Function can_deinit

Function name	can_deinit
Function prototype	void can_deinit(uint32_t can_periph);
Function descriptions	deinitialize CAN

Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize CAN0 */
can_deinit(CAN0);
```

can_software_reset

The description of can_software_reset is shown as below:

Table 3-88. Function can_software_reset

Function name	can_software_reset
Function prototype	ErrStatus can_software_reset(uint32_t can_periph);
Function descriptions	reset CAN internal state machines and CAN registers
Precondition	-
The called functions	
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;

/* reset CAN0 */
err = can_software_reset(CAN0);
```

can_init

The description of can_init is shown as below:

Table 3-89. Function can_init

Function name	can_init
----------------------	----------

Function prototype	ErrStatus can_init(uint32_t can_periph, can_parameter_struct* can_parameter_init);
Function descriptions	CAN module initialization
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
can_parameter_init	Refers to Table 3-72. Structure can_parameter_struct
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
can_parameter_struct can_parameter;
```

```
ErrStatus err;
```

```
.....
```

```
/* initialize CAN */
```

```
err = can_init(CAN0, &can_parameter);
```

can_struct_para_init

The description of can_struct_para_init is shown as below:

Table 3-90. Function can_struct_para_init

Function name	can_struct_para_init
Function prototype	void can_struct_para_init(can_struct_type_enum type, void* p_struct);
Function descriptions	initialize CAN parameter structure with a default value
Precondition	-
The called functions	-
Input parameter{in}	
type	Refers to enum Table 3-85. Enum can_struct_type_enum
Input parameter{in}	
p_struct	the pointer of the specific struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_parameter_struct can_parameter;
```

```
/* initialize CAN */
```

```
can_struct_para_init(CAN_INIT_STRUCT, &can_parameter);
```

can_private_filter_config

The description of can_private_filter_config is shown as below:

Table 3-91. Function can_private_filter_config

Function name	can_private_filter_config
Function prototype	void can_private_filter_config(uint32_t can_periph, uint32_t index, uint32_t filter_data);
Function descriptions	configure receive fifo/mailbox private filter
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0..31	CAN mailbox index selection
Input parameter{in}	
filter_data	filter data to configure
0..0xFFFFFFFF	filter data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CAN0 mailbox 0 private filter */
```

```
can_private_filter_config(CAN0, 0, 0xFFFFFFFF);
```

can_operation_mode_enter

The description of can_operation_mode_enter is shown as below:

Table 3-92. Function can_operation_mode_enter

Function name	can_operation_mode_enter
Function prototype	ErrStatus can_operation_mode_enter(uint32_t can_periph, can_operation_modes_enum mode);
Function descriptions	enter the corresponding mode
Precondition	-

The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
mode	Refers to enum Table 3-84. Enum can operation modes enum
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

```
/* CAN0 enter normal mode */
```

```
err = can_operation_mode_enter(CAN0, CAN_NORMAL_MODE);
```

can_operation_mode_get

The description of can_operation_mode_get is shown as below:

Table 3-93. Function can_operation_mode_get

Function name	can_operation_mode_get
Function prototype	can_operation_modes_enum can_operation_mode_get(uint32_t can_periph);
Function descriptions	get operation mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
can_operation_modes_enum	Refers to enum Table 3-84. Enum can operation modes enum

Example:

```
can_operation_modes_enum mode;
```

```
/* get CAN0 mode*/
```

```
mode = can_operation_mode_get(CAN0);
```


can_inactive_mode_exit

The description of can_inactive_mode_exit is shown as below:

Table 3-94. Function can_inactive_mode_exit

Function name	can_inactive_mode_exit
Function prototype	ErrStatus can_inactive_mode_exit(uint32_t can_periph);
Function descriptions	exit inactive mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

```
/* CAN0 exit INACTIVE mode */
```

```
err = can_inactive_mode_exit(CAN0);
```

can_pn_mode_exit

The description of can_pn_mode_exit is shown as below:

Table 3-95. Function can_pn_mode_exit

Function name	can_pn_mode_exit
Function prototype	ErrStatus can_pn_mode_exit(uint32_t can_periph);
Function descriptions	exit Pretended Networking mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

```
/* CAN0 exit PN mode */
```

```
err = can_pn_mode_exit(CAN0);
```

can_fd_config

The description of can_fd_config is shown as below:

Table 3-96. Function can_fd_config

Function name	can_fd_config
Function prototype	void can_fd_config(uint32_t can_periph, can_fd_parameter_struct *can_fd_para_init);
Function descriptions	can FD initialize
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
can_fd_para_init	Refers to structure Table 3-75. Structure can fd parameter struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_fd_parameter_struct fd_parameter;
```

```
/* FD parameter configurations */
```

```
.....
```

```
can_fd_config(CAN0, &fd_parameter);
```

can_bitrate_switch_enable

The description of can_bitrate_switch_enable is shown as below:

Table 3-97. Function can_bitrate_switch_enable

Function name	can_bitrate_switch_enable
Function prototype	void can_bitrate_switch_enable(uint32_t can_periph);
Function descriptions	enable bit rate switching
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CAN0 bit rate switching */
can_bitrate_switch_enable(CAN0);
```

can_bitrate_switch_disable

The description of can_bitrate_switch_disable is shown as below:

Table 3-98. Function can_bitrate_switch_disable

Function name	can_bitrate_switch_disable
Function prototype	void can_bitrate_switch_disable(uint32_t can_periph);
Function descriptions	disable bit rate switching
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CAN0 bit rate switching */
can_bitrate_switch_disable(CAN0);
```

can_tdc_get

The description of can_tdc_get is shown as below:

Table 3-99. Function can_tdc_get

Function name	can_tdc_get
Function prototype	uint32_t can_tdc_get(uint32_t can_periph);
Function descriptions	get transmitter delay compensation value
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral

<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
uint32_t	0 - 0x3F

Example:

```
uint32_t tdc;
```

```
/* get transmitter delay compensation value */
```

```
tdc = can_tdc_get(CAN0);
```

can_tdc_enable

The description of can_tdc_enable is shown as below:

Table 3-100. Function can_tdc_enable

Function name	can_tdc_enable
Function prototype	void can_tdc_enable(uint32_t can_periph);
Function descriptions	enable transmitter delay compensation
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable transmitter delay compensation */
```

```
can_tdc_enable(CAN0);
```

can_tdc_disable

The description of can_tdc_disable is shown as below:

Table 3-101. Function can_tdc_disable

Function name	can_tdc_disable
Function prototype	void can_tdc_disable(uint32_t can_periph);
Function descriptions	disable transmitter delay compensation
Precondition	-
The called functions	-

Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable transmitter delay compensation */
can_tdc_disable(CAN0);
```

can_rx_fifo_config

The description of can_rx_fifo_config is shown as below:

Table 3-102. Function can_rx_fifo_config

Function name	can_rx_fifo_config
Function prototype	void can_rx_fifo_config(uint32_t can_periph, can_fifo_parameter_struct *can_fifo_para_init);
Function descriptions	configure rx FIFO
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
can_fifo_para_init	Refers to structure Table 3-77. Structure can_fifo_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_fifo_parameter_struct fifo_struct;

/* configure rx FIFO */

.....

can_rx_fifo_config(CAN0, &fifo_struct);
```

can_rx_fifo_filter_table_config

The description of can_rx_fifo_filter_table_config is shown as below:

Table 3-103. Function `can_rx_fifo_filter_table_config`

Function name	<code>can_rx_fifo_filter_table_config</code>
Function prototype	<code>void can_rx_fifo_filter_table_config(uint32_t can_periph, can_rx_fifo_id_filter_struct id_filter_table[]);</code>
Function descriptions	configure rx FIFO filter table
Precondition	-
The called functions	-
Input parameter{in}	
<code>can_periph</code>	CAN peripheral
<code>CANx(x=0,1)</code>	CAN peripheral selection
Input parameter{in}	
<code>id_filter_table</code>	Refers to structure Table 3-76. Structure <code>can_rx_fifo_id_filter_struct</code>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_rx_fifo_id_filter_struct id_filter_table[104];

/* configure rx FIFO filter table */

.....

can_rx_fifo_filter_table_config(CAN0, id_filter_table);
```

`can_rx_fifo_read`

The description of `can_rx_fifo_read` is shown as below:

Table 3-104. Function `can_rx_fifo_read`

Function name	<code>can_rx_fifo_read</code>
Function prototype	<code>void can_rx_fifo_read(uint32_t can_periph, can_rx_fifo_struct *rx_fifo);</code>
Function descriptions	read rx FIFO data
Precondition	-
The called functions	-
Input parameter{in}	
<code>can_periph</code>	CAN peripheral
<code>CANx(x=0,1)</code>	CAN peripheral selection
Output parameter{out}	
<code>rx_fifo</code>	Refers to structure Table 3-74. Structure <code>can_rx_fifo_struct</code>
Return value	
-	-

Example:

```

can_rx_fifo_struct rx_fifo;

/* read rx FIFO data */

can_rx_fifo_read(CAN0, &rx_fifo);

```

can_rx_fifo_filter_matching_number_get

The description of can_rx_fifo_filter_matching_number_get is shown as below:

Table 3-105. Function can_rx_fifo_filter_matching_number_get

Function name	can_rx_fifo_filter_matching_number_get
Function prototype	uint32_t can_rx_fifo_filter_matching_number_get(uint32_t can_periph);
Function descriptions	get rx FIFO filter matching number
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
uint32_t	0-416

Example:

```

uint32_t number;

/* get rx FIFO filter matching number */

number = can_rx_fifo_filter_matching_number_get(CAN0);

```

can_rx_fifo_clear

The description of can_rx_fifo_clear is shown as below:

Table 3-106. Function can_rx_fifo_clear

Function name	can_rx_fifo_clear
Function prototype	void can_rx_fifo_clear(uint32_t can_periph);
Function descriptions	clear rx FIFO
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* clear rx FIFO */
```

```
can_rx_fifo_clear(CAN0);
```

can_ram_address_get

The description of can_ram_address_get is shown as below:

Table 3-107. Function can_ram_address_get

Function name	can_ram_address_get
Function prototype	uint32_t* can_ram_address_get(uint32_t can_periph, uint32_t index);
Function descriptions	get mailbox RAM address
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Output parameter{out}	
-	-
Return value	
uint32_t	0-0xFFFFFFFF

Example:

```
uint32_t address;
```

```
/* get CAN0 mailbox 0 RAM address */
```

```
address = can_ram_address_get(CAN0, 0);
```

can_mailbox_config

The description of can_mailbox_config is shown as below:

Table 3-108. Function can_mailbox_config

Function name	can_mailbox_config
Function prototype	void can_mailbox_config(uint32_t can_periph, uint32_t index, can_mailbox_descriptor_struct *mdpara);
Function descriptions	config mailbox
Precondition	-

The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure Table 3-73. Structure can_mailbox_descriptor_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_mailbox_descriptor_struct transmit_message;
```

```
.....
```

```
/* transmit message */
```

```
can_mailbox_config(CAN0, 0, &transmit_message);
```

can_mailbox_transmit_abort

The description of can_mailbox_transmit_abort is shown as below:

Table 3-109. Function can_mailbox_transmit_abort

Function name	can_mailbox_transmit_abort
Function prototype	void can_mailbox_transmit_abort(uint32_t can_periph, uint32_t index);
Function descriptions	abort mailbox transmit
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* abort mailbox transmit */
```

```
can_mailbox_transmit_abort(CAN0, 0);
```

can_mailbox_transmit_inactive

The description of can_mailbox_transmit_inactive is shown as below:

Table 3-110. Function can_mailbox_transmit_inactive

Function name	can_mailbox_transmit_inactive
Function prototype	void can_mailbox_transmit_inactive(uint32_t can_periph, uint32_t index);
Function descriptions	inactive transmit mailbox
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
index	mailbox index
<i>0-31</i>	mailbox index selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* inactive transmit mailbox */
```

```
can_mailbox_transmit_inactive(CAN0, 0);
```

can_mailbox_receive_data_read

The description of can_mailbox_receive_data_read is shown as below:

Table 3-111. Function can_mailbox_receive_data_read

Function name	can_mailbox_receive_data_read
Function prototype	ErrStatus can_mailbox_receive_data_read(uint32_t can_periph, uint32_t index, can_mailbox_descriptor_struct *mdpara);
Function descriptions	read receive mailbox data
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	

index	mailbox index
0-31	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure Table 3-73. Structure can_mailbox_descriptor_struct
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

```
can_mailbox_descriptor_struct receive_message;
```

```
.....
```

```
/* check the receive message */
```

```
err = can_mailbox_receive_data_read(CAN0, 0, &receive_message);
```

can_mailbox_receive_lock

The description of can_mailbox_receive_lock is shown as below:

Table 3-112. Function can_mailbox_receive_lock

Function name	can_mailbox_receive_lock
Function prototype	void can_mailbox_receive_lock(uint32_t can_periph, uint32_t index);
Function descriptions	lock the receive mailbox
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the receive mailbox 0 */
```

```
can_mailbox_receive_lock(CAN0, 0);
```

can_mailbox_receive_unlock

The description of can_mailbox_receive_unlock is shown as below:

Table 3-113. Function can_mailbox_receive_unlock

Function name	can_mailbox_receive_unlock
Function prototype	void can_mailbox_receive_unlock(uint32_t can_periph);
Function descriptions	unlock the receive mailbox
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the receive mailbox */
can_mailbox_receive_unlock(CAN0);
```

can_mailbox_receive_inactive

The description of can_mailbox_receive_inactive is shown as below:

Table 3-114. Function can_mailbox_receive_inactive

Function name	can_mailbox_receive_inactive
Function prototype	void can_mailbox_receive_inactive(uint32_t can_periph, uint32_t index);
Function descriptions	inactive the receive mailbox
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
index	mailbox index
<i>0-31</i>	mailbox index selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* inactive the receive mailbox */
```

```
can_mailbox_receive_inactive(CAN0, 0);
```

can_mailbox_code_get

The description of can_mailbox_code_get is shown as below:

Table 3-115. Function can_mailbox_code_get

Function name	can_mailbox_code_get
Function prototype	uint32_t can_mailbox_code_get(uint32_t can_periph, uint32_t index);
Function descriptions	get mailbox code value
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Output parameter{out}	
-	-
Return value	
uint32_t	0-0xF

Example:

```
uint32_t code;
```

```
/* get mailbox code value */
```

```
code = can_mailbox_code_get(CAN0, 0);
```

can_error_counter_config

The description of can_error_counter_config is shown as below:

Table 3-116. Function can_error_counter_config

Function name	can_error_counter_config
Function prototype	void can_error_counter_config(uint32_t can_periph, can_error_counter_struct *errcnt_struct);
Function descriptions	configure error counter
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection

Input parameter{in}	
errcnt_struct	Refers to structure Table 3-71. Structure can error counter struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_error_counter_struct err_struct;

.....

/* configure error counter */

can_error_counter_config(CAN0, &err_struct);
```

can_error_counter_get

The description of can_error_counter_get is shown as below:

Table 3-117. Function can_error_counter_get

Function name	can_error_counter_get
Function prototype	void can_error_counter_get(uint32_t can_periph, can_error_counter_struct *errcnt_struct);
Function descriptions	get error count
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
errcnt_struct	Refers to structure Table 3-71. Structure can error counter struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_error_counter_struct err_struct;

/* get error count */

can_error_counter_get(CAN0, &err_struct);
```

can_error_state_get

The description of can_error_state_get is shown as below:

Table 3-118. Function can_error_state_get

Function name	can_error_state_get
Function prototype	can_error_state_enum can_error_state_get(uint32_t can_periph);
Function descriptions	get error state indicator
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
can_error_state_enum	Refers to enum Table 3-86. Enum can_error_state_enum

Example:

```
can_error_state_enum error_state;

/* get error state indicator */

error_state = can_error_state_get(CAN0);
```

can_crc_get

The description of can_crc_get is shown as below:

Table 3-119. Function can_crc_get

Function name	can_crc_get
Function prototype	void can_crc_get(uint32_t can_periph, can_crc_struct *crc_struct);
Function descriptions	get mailbox CRC value
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Return value	
can_crc_struct	Refers to structure Table 3-80. Structure can_crc_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_crc_struct crc_struct;
```

```
/* get mailbox CRC value */
```

```
can_crc_get(CAN0, &crc_struct);
```

can_pn_mode_config

The description of can_pn_mode_config is shown as below:

Table 3-120. Function can_pn_mode_config

Function name	can_pn_mode_config
Function prototype	void can_pn_mode_config(uint32_t can_periph, can_pn_mode_config_struct *pnmod_config);
Function descriptions	configure Pretended Networking mode parameter
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Return value	
pnmod_config	Refers to structure Table 3-79. Structure can_pn_mode_config_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_pn_mode_config_struct pn_struct;
```

```
.....
```

```
/* configure Pretended Networking mode parameter */
```

```
can_pn_mode_config(CAN0, &pn_struct);
```

can_pn_mode_filter_config

The description of can_pn_mode_filter_config is shown as below:

Table 3-121. Function can_pn_mode_filter_config

Function name	can_pn_mode_filter_config
Function prototype	void can_pn_mode_filter_config(uint32_t can_periph, can_pn_mode_filter_struct *expect, can_pn_mode_filter_struct *filter);
Function descriptions	configure pn mode filter
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral

<i>CANx(x=0,1)</i>	CAN peripheral selection
Return value	
expect	Refers to structure Table 3-78. Structure can_pn_mode_filter_struct
Return value	
filter	Refers to structure Table 3-78. Structure can_pn_mode_filter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_pn_mode_filter_struct pn_filter[2];

.....

/* configure pn mode filter */

can_pn_mode_filter_config(CAN0, &pn_filter[0], &pn_filter[1]);
```

can_pn_mode_num_of_match_get

The description of can_pn_mode_num_of_match_get is shown as below:

Table 3-122. Function can_pn_mode_num_of_match_get

Function name	can_pn_mode_num_of_match_get
Function prototype	int32_t can_pn_mode_num_of_match_get(uint32_t can_periph);
Function descriptions	get matching message counter of Pretended Networking mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
int32_t	0-255 or -1

Example:

```
int32_t counter;

/* get matching message counter of Pretended Networking mode */

counter = can_pn_mode_num_of_match_get(CAN0);
```

can_pn_mode_data_read

The description of can_pn_mode_data_read is shown as below:

Table 3-123. Function can_pn_mode_data_read

Function name	can_pn_mode_data_read
Function prototype	void can_pn_mode_data_read(uint32_t can_periph, uint32_t index, can_mailbox_descriptor_struct *mdpara);
Function descriptions	get matching message
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure Table 3-73. Structure can_mailbox_descriptor_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_mailbox_descriptor_struct mb_para;
```

```
/* get matching message */
```

```
can_pn_mode_data_read(CAN0, 0, &mb_para);
```

can_self_reception_enable

The description of can_self_reception_enable is shown as below:

Table 3-124. Function can_self_reception_enable

Function name	can_self_reception_enable
Function prototype	void can_self_reception_enable(uint32_t can_periph);
Function descriptions	enable self reception
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable self reception */
can_self_reception_enable(CAN0);
```

can_self_reception_disable

The description of can_self_reception_disable is shown as below:

Table 3-125. Function can_self_reception_disable

Function name	can_self_reception_disable
Function prototype	void can_self_reception_disable(uint32_t can_periph);
Function descriptions	disable self reception
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable self reception */
can_self_reception_disable(CAN0);
```

can_transmit_abort_enable

The description of can_transmit_abort_enable is shown as below:

Table 3-126. Function can_transmit_abort_enable

Function name	can_transmit_abort_enable
Function prototype	void can_transmit_abort_enable(uint32_t can_periph);
Function descriptions	enable transmit abort
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable transmit abort */

can_transmit_abort_enable(CAN0);
```

can_transmit_abort_disable

The description of can_transmit_abort_disable is shown as below:

Table 3-127. Function can_transmit_abort_disable

Function name	can_transmit_abort_disable
Function prototype	void can_transmit_abort_disable(uint32_t can_periph);
Function descriptions	disable transmit abort
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable transmit abort */

can_transmit_abort_disable(CAN0);
```

can_auto_busoff_recovery_enable

The description of can_auto_busoff_recovery_enable is shown as below:

Table 3-128. Function can_auto_busoff_recovery_enable

Function name	can_auto_busoff_recovery_enable
Function prototype	void can_auto_busoff_recovery_enable(uint32_t can_periph);
Function descriptions	enable auto bus off recovery mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable auto bus off recovery mode */

can_auto_busoff_recovery_enable(CAN0);
```

can_auto_busoff_recovery_disable

The description of can_auto_busoff_recovery_disable is shown as below:

Table 3-129. Function can_auto_busoff_recovery_disable

Function name	can_auto_busoff_recovery_disable
Function prototype	void can_auto_busoff_recovery_disable(uint32_t can_periph);
Function descriptions	disable auto bus off recovery mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable auto bus off recovery mode */

can_auto_busoff_recovery_disable(CAN0);
```

can_time_sync_enable

The description of can_time_sync_enable is shown as below:

Table 3-130. Function can_time_sync_enable

Function name	can_time_sync_enable
Function prototype	void can_time_sync_enable(uint32_t can_periph);
Function descriptions	enable time sync mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable time sync mode */
can_time_sync_enable(CAN0);
```

can_time_sync_disable

The description of can_time_sync_disable is shown as below:

Table 3-131. Function can_time_sync_disable

Function name	can_time_sync_disable
Function prototype	void can_time_sync_disable(uint32_t can_periph);
Function descriptions	disable time sync mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable time sync mode */
can_time_sync_disable(CAN0);
```

can_edge_filter_mode_enable

The description of can_edge_filter_mode_enable is shown as below:

Table 3-132. Function can_edge_filter_mode_enable

Function name	can_edge_filter_mode_enable
Function prototype	void can_edge_filter_mode_enable(uint32_t can_periph);
Function descriptions	enable edge filter mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable edge filter mode */
```

```
can_edge_filter_mode_enable(CAN0);
```

can_edge_filter_mode_disable

The description of can_edge_filter_mode_disable is shown as below:

Table 3-133. Function can_edge_filter_mode_disable

Function name	can_edge_filter_mode_disable
Function prototype	void can_edge_filter_mode_disable(uint32_t can_periph);
Function descriptions	disable edge filter mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable edge filter mode */
```

```
can_edge_filter_mode_disable(CAN0);
```

can_ped_mode_enable

The description of can_ped_mode_enable is shown as below:

Table 3-134. Function can_ped_mode_enable

Function name	can_ped_mode_enable
Function prototype	void can_ped_mode_enable(uint32_t can_periph);
Function descriptions	enable protocol exception detection mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable protocol exception detection mode */
```

```
can_ped_mode_enable(CAN0);
```

can_ped_mode_disable

The description of can_ped_mode_disable is shown as below:

Table 3-135. Function can_ped_mode_disable

Function name	can_ped_mode_disable
Function prototype	void can_ped_mode_disable(uint32_t can_periph);
Function descriptions	disable protocol exception detection mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable protocol exception detection mode */
```

```
can_ped_mode_disable(CAN0);
```

can_arbitration_delay_bits_config

The description of can_arbitration_delay_bits_config is shown as below:

Table 3-136. Function can_arbitration_delay_bits_config

Function name	can_arbitration_delay_bits_config
Function prototype	void can_arbitration_delay_bits_config(uint32_t can_periph, uint32_t delay_bits);
Function descriptions	configure arbitration delay bits
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
delay_bits	delay bits
<i>0-31</i>	delay bits selection

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure arbitration delay bits */
can_arbitration_delay_bits_config(CAN0, 2);
```

can_bsp_mode_config

The description of can_bsp_mode_config is shown as below:

Table 3-137. Function can_bsp_mode_config

Function name	can_bsp_mode_config
Function prototype	void can_bsp_mode_config(uint32_t can_periph, uint32_t sampling_mode);
Function descriptions	configure bit sampling mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
sampling_mode	bsp sample mode
<i>CAN_BSP_MODE_ON E_SAMPLE</i>	one sample for received bit
<i>CAN_BSP_MODE_TR HEE_SAMPLES</i>	three samples for received bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure bit sampling mode */
can_bsp_mode_config(CAN0, CAN_BSP_MODE_ONE_SAMPLE);
```

can_flag_get

The description of can_flag_get is shown as below:

Table 3-138. Function can_flag_get

Function name	can_flag_get
---------------	--------------

Function prototype	FlagStatus can_flag_get(uint32_t can_periph, can_flag_enum flag);
Function descriptions	get CAN flag
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
flag	Refers to enum Table 3-82. Enum can_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus flag;
```

```
/* get CAN fifo available flag */
```

```
flag = can_flag_get(CAN0, CAN_FLAG_FIFO_AVAILABLE);
```

can_flag_clear

The description of can_flag_clear is shown as below:

Table 3-139. Function can_flag_clear

Function name	can_flag_clear
Function prototype	void can_flag_clear(uint32_t can_periph, can_flag_enum flag);
Function descriptions	clear CAN flag
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
flag	Refers to enum Table 3-82. Enum can_flag_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear CAN fifo available flag */
```

```
can_flag_clear(CAN0, CAN_FLAG_FIFO_AVAILABLE);
```

can_interrupt_enable

The description of can_interrupt_enable is shown as below:

Table 3-140. Function can_interrupt_enable

Function name	can_interrupt_enable
Function prototype	void can_interrupt_enable(uint32_t can_periph, can_interrupt_enum interrupt);
Function descriptions	enable CAN interrupt
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
interrupt	Refers to enum Table 3-81. Enum can_interrupt_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CAN bus off interrupt */
can_interrupt_enable(CAN0, CAN_INT_BUSOFF);
```

can_interrupt_disable

The description of can_interrupt_disable is shown as below:

Table 3-141. Function can_interrupt_disable

Function name	can_interrupt_disable
Function prototype	void can_interrupt_disable(uint32_t can_periph, can_interrupt_enum interrupt);
Function descriptions	disable CAN interrupt
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1)	CAN peripheral selection
Input parameter{in}	
interrupt	Refers to enum Table 3-81. Enum can_interrupt_enum
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* disable CAN bus off interrupt */
can_interrupt_disable(CAN0, CAN_INT_BUSOFF);
```

can_interrupt_flag_get

The description of can_interrupt_flag_get is shown as below:

Table 3-142. Function can_interrupt_flag_get

Function name	can_interrupt_flag_get
Function prototype	FlagStatus can_interrupt_flag_get(uint32_t can_periph, can_interrupt_flag_enum int_flag);
Function descriptions	get CAN interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection
Input parameter{in}	
int_flag	Refers to enum Table 3-83. Enum can_interrupt_flag_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus int_flag;

/* get CAN fifo available interrupt flag */
int_flag = can_interrupt_flag_get(CAN0, CAN_INT_FLAG_FIFO_AVAILABLE);
```

can_interrupt_flag_clear

The description of can_interrupt_flag_clear is shown as below:

Table 3-143. Function can_interrupt_flag_clear

Function name	can_interrupt_flag_clear
Function prototype	void can_interrupt_flag_clear(uint32_t can_periph, can_interrupt_flag_enum int_flag);
Function descriptions	clear CAN interrupt flag
Precondition	-
The called functions	-

Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0, 1)	CAN peripheral selection
Input parameter{in}	
int_flag	Refers to enum Table 3-83. Enum can interrupt flag enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear CAN fifo available interrupt flag */
```

```
can_interrupt_flag_clear(CAN0, CAN_INT_FLAG_FIFO_AVAILABLE);
```

3.5. CMP

The general purpose comparator can work either standalone(all terminal are available on I/Os) or together with the timers. It provide a trigger source when an analog signal is in a certain condition, achieves some current control by working together with a PWM output of a TIMER. The CMP registers are listed in chapter [3.5.1](#), the CMP firmware functions are introduced in chapter [3.5.2](#).

3.5.1. Descriptions of Peripheral registers

CMP registers are listed in the table shown as below:

Table 3-144. CMP registers

Registers	Descriptions
CMP_CS	CMP control and status register

3.5.2. Descriptions of Peripheral functions

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

Table 3-145. CMP firmware function

Function name	Function description
cmp_deinit	deinitialize comparator
cmp_mode_init	initialize comparator mode
cmp_output_init	initialize comparator output
cmp_outputblank_init	initialize comparator blanking function
cmp_enable	enable comparator
cmp_disable	disable comparator
cmp_voltage_scaler_enable	enable the voltage scaler

Function name	Function description
cmp_voltage_scaler_disable	disable the voltage scaler
cmp_scaler_bridge_enable	enable the scaler bridge
cmp_scaler_bridge_disable	disable the scaler bridge
cmp_lock_enable	lock the comparator
cmp_output_level_get	get output level

Enum operating_mode_enum

Table 3-146. Enum operating_mode_enum

Member name	Function description
CMP_HIGHSPEED	high speed mode
CMP_MIDDLESPEED	medium speed mode
CMP_LOWSPEED	low speed mode

Enum inverting_input_enum

Table 3-147. Enum inverting_input_enum

Member name	Function description
CMP_1_4VREFINT	VREFINT /4 input
CMP_1_2VREFINT	VREFINT /2 input
CMP_3_4VREFINT	VREFINT *3/4 input
CMP_VREFINT	VREFINT input
CMP_DAC_OUT	DAC_OUT input
CMP_IM_PC11	PC11
CMP_IM_PC10	PC10
CMP_IM_PB8	PB8
CMP_IM_PA0	PA0
CMP_IM_PA3	PA3
CMP_IM_PA4	PA4
CMP_IM_PA5	PA5
CMP_IM_PA6	PA6

Enum cmp_plus_input_enum

Table 3-148. Enum cmp_plus_input_enum

Member name	Function description
CMP_IP_PC11	PC11
CMP_IP_PC10	PC10
CMP_IP_PB8	PB8
CMP_IP_PA0	PA0
CMP_IP_PA3	PA3
CMP_IP_PA4	PA4
CMP_IP_PA5	PA5

Member name	Function description
CMP_IP_PA6	PA6

Enum cmp_hysteresis_enum

Table 3-149. Enum cmp_hysteresis_enum

Member name	Function description
CMP_HYSTERESIS_NO	output no hysteresis
CMP_HYSTERESIS_LOW	output low hysteresis
CMP_HYSTERESIS_MIDDLE	output middle hysteresis
CMP_HYSTERESIS_HIGH	output high hysteresis

Enum cmp_output_enum

Table 3-150. Enum cmp_output_enum

Member name	Function description
CMP_OUTPUT_NONE	output no selection
CMP_OUTPUT_TIMER0IC0	TIMER 0 channel0 input capture
CMP_OUTPUT_TIMER7IC0	TIMER 7 channel0 input capture

Enum cmp_output_inv_enum

Table 3-151. Enum cmp_output_inv_enum

Member name	Function description
CMP_OUTPUT_POLARITY_INVERTED	output is inverted
CMP_OUTPUT_POLARITY_NOINVERTED	output is not inverted

Enum blanking_source_enum

Table 3-152. Enum blanking_source_enum

Member name	Function description
CMP_BLANKING_NONE	output no selection
CMP_BLANKING_TIMER0_OC1	select TIMER0_CH1 as blanking source
CMP_BLANKING_TIMER7_OC1	select TIMER7_CH1 as blanking source
CMP_BLANKING_TIMER1_OC1	select TIMER1_CH1 as blanking source

Enum cmp_output_state_enum

Table 3-153. Enum cmp_output_state_enum

Member name	Function description
CMP_OUTPUTLEVEL_LOW	the output is low
CMP_OUTPUTLEVEL_HIGH	the output is high

cmp_deinit

The description of cmp_deinit is shown as below:

Table 3-154. Function cmp_deinit

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

Example:

```
/* deinitialize CMP */
cmp_deinit();
```

cmp_mode_init

The description of cmp_mode_init is shown as below:

Table 3-155. Function cmp_mode_init

Function name	cmp_mode_init
Function prototype	void cmp_mode_init(operating_mode_enum operating_mode, cmp_inverting_input_enum inverting_input, cmp_plus_input_enum plus_input, cmp_hysteresis_enum output_hysteresis);
Function descriptions	initialize comparator mode
Precondition	-
The called functions	-
Input parameter{in}	
operating_mode	operating_mode, refer to Table 3-146. Enum operating_mode_enum
Input parameter{in}	
inverting_input	inverting_input, refer to Table 3-147. Enum inverting_input_enum
Input parameter{in}	
plus_input	CMP_IP input, refer to Table 3-148. Enum cmp_plus_input_enum
Input parameter{in}	
output_hysteresis	hysteresis, refer to Table 3-149. Enum cmp_hysteresis_enum
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* initialize CMP mode */
```

```
cmp_mode_init(CMP_HIGHSPEED,CMP_1_4VREFINT, CMP_IP_PC11, CMP_HYSTERE
SIS_NO);
```

cmp_output_init

The description of cmp_output_init is shown as below:

Table 3-156. Function cmp_output_init

Function name	cmp_output_init
Function prototype	void cmp_output_init(cmp_output_enum output_selection, cmp_output_inv_enum output_polarity);
Function descriptions	initialize comparator output
Precondition	-
The called functions	-
Input parameter{in}	
output_selection	output_selection, refer to Table 3-150. Enum cmp_output_enum
Input parameter{in}	
output_polarity	output_polarity, refer to Table 3-151. Enum cmp_output_inv_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize CMP output */
```

```
cmp_output_init(CMP_OUTPUT_TIMER0IC0, CMP_OUTPUT_POLARITY_NOINVERTED);
```

cmp_outputblank_init

The description of cmp_outputblank_init is shown as below:

Table 3-157. Function cmp_outputblank_init

Function name	cmp_outputblank_init
Function prototype	void cmp_outputblank_init(blanking_source_enum blanking_source_selection);
Function descriptions	initialize comparator blanking function
Precondition	-
The called functions	-
Input parameter{in}	
blanking_source_sele	blanking source selection, refer to Table 3-152. Enum

ction	<u>blanking_source_enum</u>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize comparator blanking function */
cmp_outputblank_init(CMP_BLANKING_NONE);
```

cmp_enable

The description of cmp_enable is shown as below:

Table 3-158. Function cmp_enable

Function name	cmp_enable
Function prototype	void cmp_enable(void);
Function descriptions	enable comparator
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CMP */
cmp_enable();
```

cmp_disable

The description of cmp_disable is shown as below:

Table 3-159. Function cmp_disable

Function name	cmp_disable
Function prototype	void cmp_disable(void);
Function descriptions	disable comparator
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CMP */

cmp_disable();
```

cmp_voltage_scaler_enable

The description of cmp_voltage_scaler_enable is shown as below:

Table 3-160. Function cmp_voltage_scaler_enable

Function name	cmp_voltage_scaler_enable
Function prototype	void cmp_voltage_scaler_enable(void);
Function descriptions	enable comparator the voltage scaler
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CMP the voltage scaler */

cmp_voltage_scaler_enable();
```

cmp_voltage_scaler_disable

The description of cmp_voltage_scaler_disable is shown as below:

Table 3-161. Function cmp_voltage_scaler_disable

Function name	cmp_voltage_scaler_disable
Function prototype	void cmp_voltage_scaler_disable(void);
Function descriptions	disable comparator the voltage scaler
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CMP the voltage scaler */

cmp_voltage_scaler_disable();
```

cmp_scaler_bridge_enable

The description of cmp_scaler_bridge_enable is shown as below:

Table 3-162. Function cmp_scaler_bridge_enable

Function name	cmp_scaler_bridge_enable
Function prototype	void cmp_scaler_bridge_enable(void);
Function descriptions	enable comparator the scaler bridge
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CMP the scaler bridge */
cmp_scaler_bridge_enable();
```

cmp_scaler_bridge_disable

The description of cmp_scaler_bridge_disable is shown as below:

Table 3-163. Function cmp_scaler_bridge_disable

Function name	cmp_scaler_bridge_disable
Function prototype	void cmp_scaler_bridge_disable(void);
Function descriptions	disable comparator the scaler bridge
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CMP the scaler bridge */
cmp_scaler_bridge_disable();
```

cmp_lock_enable

The description of cmp_lock_enable is shown as below:

Table 3-164. Function cmp_lock_enable

Function name	cmp_lock_enable
----------------------	-----------------

Function prototype	void cmp_lock_enable(void);
Function descriptions	lock the comparator
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock CMP register */
```

```
cmp_lock_enable();
```

cmp_output_level_get

The description of cmp_output_level_get is shown as below:

Table 3-165. Function cmp_output_level_get

Function name	cmp_output_level_get
Function prototype	cmp_output_state_enum cmp_output_level_get(void);
Function descriptions	get output level
Precondition	-
The called functions	-
Output parameter{out}	
-	-
Return value	
cmp_output_state_enum	the output level, refer to Table 3-153. Enum cmp_output_state_enum

Example:

```
uint32_t level;
```

```
/* get CMP output level */
```

```
level = cmp_output_level_get();
```

3.6. 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 [3.6.1](#), the CRC firmware functions are introduced in chapter [3.6.2](#).

3.6.1. Descriptions of Peripheral registers

CRC registers are listed in the table shown as below:

Table 3-166. CRC Registers

Registers	Descriptions
CRC_DATA	CRC data register
CRC_FDATA	CRC free data register
CRC_CTL	CRC control register
CRC_IDATA	CRC initialization data register
CRC_POLY	CRC polynomial register

3.6.2. Descriptions of Peripheral functions

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

Table 3-167. CRC firmware function

Function name	Function description
crc_deinit	deinit CRC calculation unit
crc_reverse_output_data_enable	enable the reverse operation of output data
crc_reverse_output_data_disable	disable the reverse operation of output data
crc_data_register_reset	reset data register to the value of initializaiton data register
crc_data_register_read	read the data register
crc_free_data_register_read	read the free data register
crc_free_data_register_write	write the free data register
crc_init_data_register_write	write the initialization value register
crc_input_data_reverse_config	configure the CRC input data function
crc_polynomial_size_set	configure the CRC size of polynomial function
crc_polynomial_set	configure the CRC polynomial value function
crc_single_data_calculate	CRC calculate single data
crc_block_data_calculate	CRC calculate a data array

crc_deinit

The description of crc_deinit is shown as below:

Table 3-168. 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_reverse_output_data_enable

The description of crc_reverse_output_data_enable is shown as below:

Table 3-169. Function crc_reverse_output_data_enable

Function name	crc_reverse_output_data_enable
Function prototype	void crc_reverse_output_data_enable(void);
Function descriptions	enable the reverse operation of output data
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CRC reverse operation of output data */
crc_reverse_output_data_enable();
```

crc_reverse_output_data_disable

The description of crc_reverse_output_data_disable is shown as below:

Table 3-170. Function crc_reverse_output_data_disable

Function name	crc_reverse_output_data_disable
Function prototype	void crc_reverse_output_data_disable(void);
Function descriptions	disable the reverse operation of output data
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* disable crc reverse operation of output data */
```

```
crc_reverse_output_data_disable();
```

crc_data_register_reset

The description of crc_data_register_reset is shown as below:

Table 3-171. Function crc_data_register_reset

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

Example:

```
/* reset crc data register */
```

```
crc_data_register_reset();
```

crc_data_register_read

The description of crc_data_register_read is shown as below:

Table 3-172. Function crc_data_register_read

Function name	crc_data_register_read
Function prototype	uint32_t crc_data_register_read(void);
Function descriptions	read 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 3-173. 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 free data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	8-bit value of the free data register (0-0xFF)

Example:

```
/* read crc free data register */

uint8_t crc_value = 0;

crc_value = crc_free_data_register_read();
```

crc_free_data_register_write

The description of `crc_free_data_register_write` is shown as below:

Table 3-174. 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 the free data register
Precondition	-
The called functions	-
Input parameter{in}	
free_data	specify 8-bit data
Output parameter{out}	

-	-
Return value	
-	-

Example:

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

crc_init_data_register_write

The description of `crc_init_data_register_write` is shown as below:

Table 3-175. Function `crc_init_data_register_write`

Function name	<code>crc_init_data_register_write</code>
Function prototype	<code>void crc_init_data_register_write(uint32_t init_data)</code>
Function descriptions	write the initializaiton data register
Precondition	-
The called functions	-
Input parameter{in}	
init_data	specify 32-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write crc initializaiton data register */
crc_init_data_register_write(0x11223344);
```

crc_input_data_reverse_config

The description of `crc_input_data_reverse_config` is shown as below:

Table 3-176. Function `crc_input_data_reverse_config`

Function name	<code>crc_input_data_reverse_config</code>
Function prototype	<code>void crc_input_data_reverse_config(uint32_t data_reverse)</code>
Function descriptions	configure the crc input data function
Precondition	-
The called functions	-
Input parameter{in}	
data_reverse	specify input data reverse function
<code>CRC_INPUT_DATA_NOT</code>	input data is not reversed

<i>CRC_INPUT_DATA_BYTE</i>	input data is reversed on 8 bits
<i>CRC_INPUT_DATA_HALFWORD</i>	input data is reversed on 16 bits
<i>CRC_INPUT_DATA_WORD</i>	input data is reversed on 32 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the crc input data */
```

```
crc_input_data_reverse_config(CRC_INPUT_DATA_WORD);
```

crc_polynomial_size_set

The description of `crc_polynomial_size_set` is shown as below:

Table 3-177. Function `crc_polynomial_size_set`

Function name	<code>crc_polynomial_size_set</code>
Function prototype	<code>void crc_polynomial_size_set(uint32_t poly_size)</code>
Function descriptions	configure the CRC size of polynomial function
Precondition	-
The called functions	-
Input parameter{in}	
poly_size	size of polynomial
<i>CRC_CTL_PS_32</i>	32-bit polynomial for CRC calculation
<i>CRC_CTL_PS_16</i>	16-bit polynomial for CRC calculation
<i>CRC_CTL_PS_8</i>	8-bit polynomial for CRC calculation
<i>CRC_CTL_PS_7</i>	7-bit polynomial for CRC calculation
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CRC polynomial size */
```

```
crc_polynomial_size_set(CRC_CTL_PS_7);
```

crc_polynomial_set

The description of `crc_polynomial_set` is shown as below:

Table 3-178. Function `crc_polynomial_set`

Function name	<code>crc_polynomial_set</code>
Function prototype	<code>void crc_polynomial_set(uint32_t poly)</code>
Function descriptions	configure the CRC polynomial value function
Precondition	-
The called functions	-
Input parameter{in}	
poly	configurable polynomial value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CRC polynomial value */
```

```
crc_polynomial_set(0x11223344);
```

`crc_single_data_calculate`

The description of `crc_single_data_calculate` is shown as below:

Table 3-179. Function `crc_single_data_calculate`

Function name	<code>crc_single_data_calculate</code>
Function prototype	<code>uint32_t crc_single_data_calculate(uint32_t sdata, uint8_t data_format);</code>
Function descriptions	CRC calculate single data
Precondition	-
The called functions	-
Input parameter{in}	
sdata	specify input data
Input parameter{in}	
data_format	input data format
<code>INPUT_FORMAT_WORD</code>	input data in word format
<code>INPUT_FORMAT_HALFWORD</code>	input data in half-word format
<code>INPUT_FORMAT_BYTE</code>	input data in byte format
Output parameter{out}	
-	-
Return value	
uint32_t	CRC calculate value (0-0xFFFFFFFF)

Example:

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

```
uint32_t val = 0, valcrc = 0;
```

```
val = (uint32_t)0xabcd1234;
```

```
valcrc = crc_single_data_calculate(val, INPUT_FORMAT_WORD);
```

crc_block_data_calculate

The description of `crc_block_data_calculate` is shown as below:

Table 3-180. Function `crc_block_data_calculate`

Function name	<code>crc_block_data_calculate</code>
Function prototype	<code>uint32_t crc_block_data_calculate(void *array, uint32_t size, uint8_t data_format);</code>
Function descriptions	CRC calculate a data array
Precondition	-
The called functions	-
Input parameter{in}	
array	pointer to the input data array
Input parameter{in}	
size	size of the array
Input parameter{in}	
data_format	input data format
<code>INPUT_FORMAT_WORD</code>	input data in word format
<code>INPUT_FORMAT_HALFWORD</code>	input data in half-word format
<code>INPUT_FORMAT_BYTE</code>	input data in byte format
Output parameter{out}	
-	-
Return value	
uint32_t	CRC calculate value (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};
```

```
valcrc = crc_block_data_calculate(data_buffer, BUFFER_SIZE, INPUT_FORMAT_WORD);
```

3.7. DBG

The DBG hold unit helps debugger to debug power saving mode. The DBG registers are listed in chapter [3.7.1](#). the DBG firmware functions are introduced in chapter [3.7.2](#).

3.7.1. Descriptions of Peripheral registers

DBG registers are listed in the table shown as below:

Table 3-181. DBG Registers

Registers	Descriptions
DBG_ID	DBG ID code register
DBG_CTL	DBG control register

3.7.2. Descriptions of Peripheral functions

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

Table 3-182. DBG firmware function

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

Enum dbg_periph_enum

Table 3-183. Enum dbg_periph_enum

Member name	Function description
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_TIMER0_HOLD	hold TIMER0 counter when core is halted
DBG_TIMER1_HOLD	hold TIMER1 counter when core is halted
DBG_I2C0_HOLD	hold I2C0 smbus when core is halted
DBG_I2C1_HOLD	hold I2C1 smbus when core is halted
DBG_TIMER7_HOLD	hold TIMER7 counter when core is halted
DBG_TIMER5_HOLD	hold TIMER5 counter when core is halted
DBG_TIMER6_HOLD	hold TIMER6 counter when core is halted
DBG_MFCOM_HOLD	hold MFCOM counter when core is halted
DBG_CAN0_HOLD	hold CAN0 counter when core is halted
DBG_CAN1_HOLD	hold CAN1 counter when core is halted

Member name	Function description
DBG_TIMER20_HOLD	hold TIMER20 counter when core is halted
DBG_TIMER19_HOLD	hold TIMER19 counter when core is halted

dbg_deinit

The description of dbg_deinit is shown as below:

Table 3-184. 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 3-185. 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 3-186. 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
<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:

```
/* enable low power behavior when the mcu is in debug mode */
```

```
dbg_low_power_enable(DBG_LOW_POWER_SLEEP);
```

dbg_low_power_disable

The description of dbg_low_power_disable is shown as below:

Table 3-187. 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>	do not keep debugger connection during sleep mode

<i>DBG_LOW_POWER_D EEPSLEEP</i>	do not keep debugger connection during deepsleep mode
<i>DBG_LOW_POWER_S TANDBY</i>	do not keep debugger connection during standby mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable low power behavior when the mcu is in debug mode */
```

```
dbg_low_power_disable(DBG_LOW_POWER_SLEEP);
```

dbg_periph_enable

The description of dbg_periph_enable is shown as below:

Table 3-188. 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 3-183. Enum dbg_periph_enum
<i>DBG_FWDGT_HOLD</i>	hold FWDGT counter when core is halted
<i>DBG_WWDGT_HOLD</i>	hold WWDGT counter when core is halted
<i>DBG_TIMERx_HOLD</i>	x=1,5,6,7,19,20 hold TIMEx counter when core is halted
<i>DBG_I2Cx_HOLD</i>	x=0,1 hold I2Cx smbus when core is halted
<i>DBG_MFCOM_HOLD</i>	hold MFCOM counter when core is halted
<i>DBG_CANx_HOLD</i>	x=0,1 hold CANx counter when core is halted
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable peripheral behavior when the mcu is in debug mode */
```

```
dbg_periph_enable(DBG_TIMER1_HOLD);
```

dbg_periph_disable

The description of dbg_periph_disable is shown as below:

Table 3-189. 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 3-183. Enum dbg_periph_enum
DBG_FWDGT_HOLD	hold FWDGT counter when core is halted
DBG_WWDGT_HOLD	hold WWDGT counter when core is halted
DBG_TIMERx_HOLD	x=1,5,6,7,19,20 hold TIMERx counter when core is halted
DBG_I2Cx_HOLD	x=0,1 hold I2Cx smbus when core is halted
DBG_MFCOM_HOLD	hold MFCOM counter when core is halted
DBG_CANx_HOLD	x=0,1 hold CANx counter when core is halted
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable peripheral behavior when the mcu is in debug mode */
```

```
dbg_periph_disable(DBG_TIMER1_HOLD);
```

3.8. DAC

The Digital-to-analog converter converts 12-bit digital data to a voltage on the external pins. The DAC registers are listed in chapter [3.8.1](#), the DAC firmware functions are introduced in chapter [3.8.2](#).

3.8.1. Descriptions of Peripheral registers

DAC registers are listed in the table shown as below:

Table 3-190. DAC Registers

Registers	Descriptions
DAC_CTL	DAC control register
DAC_SWT	DAC software trigger register
OUT_R12DH	DAC_OUT 12-bit right-aligned data holding register
OUT_L12DH	DAC_OUT 12-bit left-aligned data holding register
OUT_R8DH	DAC_OUT 8-bit right-aligned data holding register
OUT_DO	DAC_OUT data output register
DAC_STAT	DAC status register

3.8.2. Descriptions of Peripheral functions

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

Table 3-191. DAC firmware function

Function name	Function description
<code>dac_deinit</code>	deinitialize DAC
<code>dac_enable</code>	enable DAC
<code>dac_disable</code>	disable DAC
<code>dac_dma_enable</code>	enable DAC DMA
<code>dac_dma_disable</code>	disable DAC DMA
<code>dac_output_buffer_enable</code>	enable DAC output buffer
<code>dac_output_buffer_disable</code>	disable DAC output buffer
<code>dac_output_value_get</code>	get DAC output value
<code>dac_data_set</code>	set DAC data holding register value
<code>dac_trigger_enable</code>	enable DAC trigger
<code>dac_trigger_disable</code>	disable DAC trigger
<code>dac_trigger_source_config</code>	configure DAC trigger source
<code>dac_software_trigger_enable</code>	enable DAC software trigger
<code>dac_software_trigger_disable</code>	disable DAC software trigger
<code>dac_wave_mode_config</code>	configure DAC wave mode
<code>dac_wave_bit_width_config</code>	configure DAC wave bit width
<code>dac_lfsr_noise_config</code>	configure DAC LFSR noise mode
<code>dac_triangle_noise_config</code>	configure DAC triangle noise mode
<code>dac_flag_get</code>	get DAC flag
<code>dac_flag_clear</code>	clear DAC flag
<code>dac_interrupt_enable</code>	enable DAC interrupt
<code>dac_interrupt_disable</code>	disable DAC interrupt
<code>dac_interrupt_flag_get</code>	get DAC interrupt flag
<code>dac_interrupt_flag_clear</code>	clear DAC interrupt flag

`dac_deinit`

The description of `dac_deinit` is shown as below:

Table 3-192. Function `dac_deinit`

Function name	<code>dac_deinit</code>
Function prototype	<code>void dac_deinit(void);</code>
Function descriptions	deinitialize DAC
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:

```
/* deinitialize DAC */
```

```
dac_deinit();
```

dac_enable

The description of dac_enable is shown as below:

Table 3-193. Function dac_enable

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

Example:

```
/* enable DAC */
```

```
dac_enable();
```

dac_disable

The description of dac_disable is shown as below:

Table 3-194. Function dac_disable

Function name	dac_disable
Function prototype	void dac_disable(void);
Function descriptions	disable DAC
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* disable DAC */
```

```
dac_disable();
```

dac_dma_enable

The description of dac_dma_enable is shown as below:

Table 3-195. Function dac_dma_enable

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

Example:

```
/* enable DAC DMA function */
```

```
dac_dma_enable();
```

dac_dma_disable

The description of dac_dma_disable is shown as below:

Table 3-196. Function dac_dma_disable

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

-	-
---	---

Example:

```
/* disable DAC DMA function */
dac_dma_disable();
```

dac_output_buffer_enable

The description of dac_output_buffer_enable is shown as below:

Table 3-197. Function dac_output_buffer_enable

Function name	dac_output_buffer_enable
Function prototype	void dac_output_buffer_enable(void);
Function descriptions	enable DAC output buffer
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC output buffer */
dac_output_buffer_enable();
```

dac_output_buffer_disable

The description of dac_output_buffer_disable is shown as below:

Table 3-198. Function dac_output_buffer_disable

Function name	dac_output_buffer_disable
Function prototype	void dac_output_buffer_disable(void);
Function descriptions	disable DAC output buffer
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC output buffer */
dac_output_buffer_disable();
```

dac_output_value_get

The description of dac_output_value_get is shown as below:

Table 3-199. Function dac_output_value_get

Function name	dac_output_value_get
Function prototype	uint16_t dac_output_value_get(void);
Function descriptions	get DAC output value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	DAC output data (0~4095)

Example:

```
/* get DAC output value */
uint16_t data = 0;
data = dac_output_value_get();
```

dac_data_set

The description of dac_data_set is shown as below:

Table 3-200. Function dac_data_set

Function name	dac_data_set
Function prototype	void dac_data_set(uint32_t dac_align, uint16_t data);
Function descriptions	set DAC data holding register value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Input parameter{in}	
dac_align	DAC align mode
DAC_ALIGN_12B_R	data right 12b alignment
DAC_ALIGN_12B_L	data left 12b alignment

<i>DAC_ALIGN_8B_R</i>	data right 8b alignment
Input parameter{in}	
data	data to be loaded, 0~4095
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the DAC specified data holding register value */
```

```
dac_data_set(DAC_ALIGN_8B_R,0xff);
```

dac_trigger_enable

The description of `dac_trigger_enable` is shown as below:

Table 3-201. Function `dac_trigger_enable`

Function name	<code>dac_trigger_enable</code>
Function prototype	<code>void dac_trigger_enable(void);</code>
Function descriptions	enable DAC trigger
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC trigger */
```

```
dac_trigger_enable();
```

dac_trigger_disable

The description of `dac_trigger_disable` is shown as below:

Table 3-202. Function `dac_trigger_disable`

Function name	<code>dac_trigger_disable</code>
Function prototype	<code>void dac_trigger_disable(void);</code>
Function descriptions	disable DAC trigger
Precondition	-
The called functions	-

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

Example:

```
/* disable DAC trigger */
```

```
dac_trigger_disable();
```

dac_trigger_source_config

The description of dac_trigger_source_config is shown as below:

Table 3-203. Function dac_trigger_source_config

Function name	dac_trigger_source_config
Function prototype	void dac_trigger_source_config(uint32_t triggersource);
Function descriptions	configure DAC trigger source
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Input parameter{in}	
triggersource	external triggers of DAC
DAC_TRIGGER_EXTRIG	TRIGSEL trigger
DAC_TRIGGER_SOFTWARE	software trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC trigger source*/
```

```
dac_trigger_source_config(DAC_TRIGGER_SOFTWARE);
```

dac_software_trigger_enable

The description of dac_software_trigger_enable is shown as below:

Table 3-204. Function dac_software_trigger_enable

Function name	dac_software_trigger_enable
Function prototype	void dac_software_trigger_enable(void);
Function descriptions	enable DAC software trigger
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC software trigger */
dac_software_trigger_enable();
```

dac_software_trigger_disable

The description of dac_software_trigger_disable is shown as below:

Table 3-205. Function dac_software_trigger_disable

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

Example:

```
/* disable DAC software trigger */
dac_software_trigger_disable();
```

dac_wave_mode_config

The description of dac_wave_mode_config is shown as below:

Table 3-206. Function `dac_wave_mode_config`

Function name	<code>dac_wave_mode_config</code>
Function prototype	<code>void dac_wave_mode_config(uint32_t wave_mode);</code>
Function descriptions	configure DAC wave mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Input parameter{in}	
wave_mode	DAC wave mode
<code>DAC_WAVE_DISABLE</code>	wave mode disable
<code>DAC_WAVE_MODE_LFSR</code>	LFSR noise mode
<code>DAC_WAVE_MODE_TRIANGLE</code>	triangle noise mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC wave mode */
```

```
dac_wave_mode_config(DAC_WAVE_DISABLE);
```

`dac_wave_bit_width_config`

The description of `dac_wave_bit_width_config` is shown as below:

Table 3-207. Function `dac_wave_bit_width_config`

Function name	<code>dac_wave_bit_width_config</code>
Function prototype	<code>void dac_wave_bit_width_config(uint32_t bit_width);</code>
Function descriptions	configure DAC wave bit width
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Input parameter{in}	
bit_width	DAC noise wave bit width
<code>DAC_WAVE_BIT_WIDTH_12</code>	$x = 1..12$
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* configure DAC wave bit width */
```

```
dac_wave_bit_width_config(DAC_WAVE_BIT_WIDTH_1);
```

dac_lfsr_noise_config

The description of dac_lfsr_noise_config is shown as below:

Table 3-208. Function dac_lfsr_noise_config

Function name	dac_lfsr_noise_config
Function prototype	void dac_lfsr_noise_config(uint32_t unmask_bits);
Function descriptions	configure DAC LFSR noise mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Input parameter{in}	
unmask_bits	LFSR noise unmask bits
<i>DAC_LFSR_BIT0</i>	unmask the LFSR bit0
<i>DAC_LFSR_BITSx_0</i>	unmask the LFSR bits[x:0],x=1..11
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC LFSR noise mode */
```

```
dac_lfsr_noise_config(DAC_LFSR_BIT0);
```

dac_triangle_noise_config

The description of dac_triangle_noise_config is shown as below:

Table 3-209. Function dac_triangle_noise_config

Function name	dac_triangle_noise_config
Function prototype	void dac_triangle_noise_config(uint32_t amplitude);
Function descriptions	configure DAC triangle noise mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Input parameter{in}	
amplitude	the amplitude of triangle noise
<i>DAC_TRIANGLE_AMP</i> <i>LITUDE_x</i>	$x = 2^n - 1 (n = 1..12)$
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC triangle noise mode */
dac_triangle_noise_config(DAC_TRIANGLE_AMPLITUDE_1);
```

dac_flag_get

The description of `dac_flag_get` is shown as below:

Table 3-210. Function `dac_flag_get`

Function name	<code>dac_flag_get</code>
Function prototype	<code>FlagStatus dac_flag_get(uint32_t flag);</code>
Function descriptions	get DAC flag
Precondition	-
The called functions	-
Input parameter{in}	
dac_flag	DAC flag
<i>DAC_FLAG_DDUDR</i>	DMA underrun flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the specified DAC flag */
FlagStatus dac_falg = RESET;
dac_falg = dac_flag_get(DAC_FLAG_DDUDR);
```

dac_flag_clear

The description of `dac_flag_clear` is shown as below:

Table 3-211. Function `dac_flag_clear`

Function name	<code>dac_flag_clear</code>
Function prototype	<code>void dac_flag_clear(uint32_t flag);</code>

Function descriptions	clear DAC flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	DAC flag
<i>DAC_FLAG_DDUDR</i>	DMA underrun flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the specified DAC flag(DAC DMA underrun flag)*/
dac_flag_clear(DAC_FLAG_DDUDR);
```

dac_interrupt_enable

The description of `dac_interrupt_enable` is shown as below:

Table 3-212. Function `dac_interrupt_enable`

Function name	<code>dac_interrupt_enable</code>
Function prototype	<code>void dac_interrupt_enable(uint32_t interrupt);</code>
Function descriptions	enable DAC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the DAC interrupt
<i>DAC_INT_DDUDRIE</i>	DMA underrun interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC interrupt(DAC DMA underrun interrupt) */
dac_interrupt_enable(DAC_INT_DDUDRIE);
```

dac_interrupt_disable

The description of `dac_interrupt_disable` is shown as below:

Table 3-213. Function `dac_interrupt_disable`

Function name	<code>dac_interrupt_disable</code>
----------------------	------------------------------------

Function prototype	void dac_interrupt_disable(uint32_t interrupt);
Function descriptions	disable DAC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the DAC interrupt
<i>DAC_INT_DDUDRIE</i>	DMA underrun interrupt disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC interrupt(DAC DMA underrun interrupt) */
```

```
dac_interrupt_disable(DAC_INT_DDUDRIE);
```

dac_interrupt_flag_get

The description of dac_interrupt_flag_get is shown as below:

Table 3-214. Function dac_interrupt_flag_get

Function name	dac_interrupt_flag_get
Function prototype	FlagStatus dac_interrupt_flag_get(uint32_t int_flag);
Function descriptions	get DAC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	DAC interrupt flag
<i>DAC_INT_FLAG_DDU DR</i>	DMA underrun interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	the state of DAC
<i>SET</i>	DMA underrun error interrupt occurred
<i>RESET</i>	No DMA underrun error interrupt occurred

Example:

```
/* get the specified DAC interrupt flag(DAC DMA underrun interrupt flag) */
```

```
FlagStatus dac_falg = RESET;
```

```
dac_falg = dac_interrupt_flag_get(DAC_INT_FLAG_DDUDR);
```

dac_interrupt_flag_clear

The description of dac_interrupt_flag_clear is shown as below:

Table 3-215. Function dac_interrupt_flag_clear

Function name	dac_interrupt_flag_clear
Function prototype	void dac_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear the specified DAC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	DAC interrupt flag
<i>DAC_INT_FLAG_DDU DR</i>	DMA underrun interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the specified DAC interrupt flag(DAC DMA underrun interrupt flag)*/
```

```
dac_interrupt_flag_clear(DAC_INT_FLAG_DDUDR);
```

3.9. DMA / DMAMUX

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 [3.9.1](#), the DMA firmware functions are introduced in chapter [3.9.2](#).

DMAMUX is a transmission scheduler for DMA requests. The DMAMUX request multiplexer is used for routing a DMA request line between the peripherals / generated DMA request (from the DMAMUX request generator) and the DMA controller. The DMAMUX registers are listed in chapter [3.9.1](#), the DMAMUX firmware functions are introduced in chapter [3.9.2](#).

3.9.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

Table 3-216. DMA Registers

Registers	Descriptions
DMA_INTF	Interrupt flag register
DMA_INTC	Interrupt flag clear register

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

DMAMUX registers are listed in the table shown as below:

Table 3-217. DMAMUX Registers

Registers	Descriptions
DMAMUX_RM_CHx CFG (x=0..11)	Request multiplexer channel x configuration register
DMAMUX_RM_INT F	Request multiplexer channel interrupt flag register
DMAMUX_RM_INT C	Request multiplexer channel interrupt flag clear register
DMAMUX_RG_CHx CFG (x=0..3)	Request generator channel x configuration register
DMAMUX_RG_INT F	Request generator channel interrupt flag register
DMAMUX_RG_INT C	Request generator channel interrupt flag clear register

3.9.2. Descriptions of Peripheral functions

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

Table 3-218. DMA firmware function

Function name	Function description
dma_deinit	deinitialize DMA a channel registers
dma_struct_para_init	initialize the parameters of DMA struct with the default values
dma_init	initialize DMA channel
dma_circulation_enable	enable DMA circulation mode
dma_circulation_disable	disable DMA circulation mode
dma_memory_to_memory_enable	enable memory to memory mode
dma_memory_to_memory_disable	disable memory to memory mode
dma_channel_enable	enable DMA channel
dma_channel_disable	disable DMA channel
dma_periph_address_config	set DMA peripheral base address
dma_memory_address_config	set DMA memory base address

Function name	Function description
dma_transfer_number_config	set the number of remaining data to be transferred by the DMA
dma_transfer_number_get	get the number of remaining data to be transferred by the DMA
dma_priority_config	configure priority level of DMA channel
dma_memory_width_config	configure transfer data size of memory
dma_periph_width_config	configure transfer data size of peripheral
dma_memory_increase_enable	enable next address increasement algorithm of memory
dma_memory_increase_disable	disable next address increasement algorithm of memory
dma_periph_increase_enable	enable next address increasement algorithm of peripheral
dma_periph_increase_disable	disable next address increasement algorithm of peripheral
dma_transfer_direction_config	configure the direction of data transfer on the channel
dma_flag_get	check DMA flag is set or not
dma_flag_clear	clear DMA a channel flag
dma_interrupt_enable	enable DMA interrupt
dma_interrupt_disable	disable DMA interrupt
dma_interrupt_flag_get	check DMA flag and interrupt enable bit is set or not
dma_interrupt_flag_clear	clear DMA a channel flag

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

Table 3-219. DMAMUX firmware function

Function name	Function description
dmamux_sync_struct_para_init	initialize the parameters of DMAMUX synchronization mode structure with the default values
dmamux_synchronization_init	initialize DMAMUX request multiplexer channel synchronization mode
dmamux_synchronization_enable	enable synchronization mode
dmamux_synchronization_disable	disable synchronization mode
dmamux_event_generation_enable	enable event generation
dmamux_event_generation_disable	disable event generation
dmamux_gen_struct_para_init	initialize the parameters of DMAMUX request generator structure with the default values
dmamux_request_generator_init	initialize DMAMUX request generator channel
dmamux_request_generator_channel_enable	enable DMAMUX request generator channel
dmamux_request_generator_channel_disable	disable DMAMUX request generator channel
dmamux_synchronization_polarity_config	configure synchronization input polarity
dmamux_request_forward_number_config	configure number of DMA requests to forward

Function name	Function description
dmamux_sync_id_config	configure synchronization input identification
dmamux_request_id_config	configure multiplexer input identification
dmamux_trigger_polarity_config	configure trigger input polarity
dmamux_request_generate_number_config	configure number of DMA requests to be generated
dmamux_trigger_id_config	configure trigger input identification
dmamux_flag_get	get DMAMUX flag
dmamux_flag_clear	clear DMAMUX flag
dmamux_interrupt_enable	enable DMAMUX interrupt
dmamux_interrupt_disable	disable DMAMUX interrupt
dmamux_interrupt_flag_get	get DMAMUX interrupt flag
dmamux_interrupt_flag_clear	clear DMAMUX interrupt flag

Structure dma_parameter_struct

Table 3-220. Structure dma_parameter_struct

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

Structure dmamux_sync_parameter_struct

Table 3-221. Structure dmamux_sync_parameter_struct

Member name	Function description
sync_id	synchronization input identification
sync_polarity	synchronization input polarity
request_number	number of DMA requests to forward

Structure dmamux_gen_parameter_struct

Table 3-222. Structure dmamux_gen_parameter_struct

Member name	Function description
trigger_id	trigger input identification
trigger_polarity	DMAMUX request generator trigger polarity
request_number	number of DMA requests to be generated

Enum dma_channel_enum

Table 3-223. Enum dma_channel_enum

Member name	Function description
DMA_CH0	DMA Channel 0
DMA_CH1	DMA Channel 1
DMA_CH2	DMA Channel 2
DMA_CH3	DMA Channel 3
DMA_CH4	DMA Channel 4
DMA_CH5	DMA Channel 5
DMA_CH6	DMA Channel 6

Enum dmamux_multiplexer_channel_enum

Table 3-224. Enum dmamux_multiplexer_channel_enum

Member name	Function description
DMAMUX_MUXCH 0	DMAMUX request multiplexer Channel 0
DMAMUX_MUXCH 1	DMAMUX request multiplexer Channel 1
DMAMUX_MUXCH 2	DMAMUX request multiplexer Channel 2
DMAMUX_MUXCH 3	DMAMUX request multiplexer Channel 3
DMAMUX_MUXCH 4	DMAMUX request multiplexer Channel 4
DMAMUX_MUXCH 5	DMAMUX request multiplexer Channel 5
DMAMUX_MUXCH 6	DMAMUX request multiplexer Channel 6
DMAMUX_MUXCH 7	DMAMUX request multiplexer Channel 7
DMAMUX_MUXCH 8	DMAMUX request multiplexer Channel 8
DMAMUX_MUXCH 9	DMAMUX request multiplexer Channel 9
DMAMUX_MUXCH 10	DMAMUX request multiplexer Channel 10
DMAMUX_MUXCH 11	DMAMUX request multiplexer Channel 11

Enum dmamux_generator_channel_enum

Table 3-225. Enum dmamux_generator_channel_enum

Member name	Function description
DMAMUX_GENCH0	DMAMUX request generator Channel0
DMAMUX_GENCH1	DMAMUX request generator Channel1
DMAMUX_GENCH2	DMAMUX request generator Channel2
DMAMUX_GENCH3	DMAMUX request generator Channel3

Enum dmamux_interrupt_enum

Table 3-226. Enum dmamux_interrupt_enum

Member name	Function description
DMAMUX_INT_MU XCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
DMAMUX_INT_MU XCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
DMAMUX_INT_MU XCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun interrupt
DMAMUX_INT_MU XCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
DMAMUX_INT_MU XCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun interrupt
DMAMUX_INT_MU XCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun interrupt
DMAMUX_INT_MU XCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun interrupt
DMAMUX_INT_MU XCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun interrupt
DMAMUX_INT_MU XCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun interrupt
DMAMUX_INT_MU XCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun interrupt
DMAMUX_INT_MU XCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun interrupt
DMAMUX_INT_MU XCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun interrupt
DMAMUX_INT_GE NCH0_TO	DMAMUX request generator channel 0 trigger overrun interrupt
DMAMUX_INT_GE NCH1_TO	DMAMUX request generator channel 1 trigger overrun interrupt
DMAMUX_INT_GE NCH2_TO	DMAMUX request generator channel 2 trigger overrun interrupt

DMAMUX_INT_GE NCH3_TO	DMAMUX request generator channel 3 trigger overrun interrupt
--------------------------	--

Enum dmamux_flag_enum

Table 3-227. Enum dmamux_flag_enum

Member name	Function description
DMAMUX_FLAG_MUXCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun flag
DMAMUX_FLAG_MUXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun flag
DMAMUX_FLAG_MUXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun flag
DMAMUX_FLAG_MUXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun flag
DMAMUX_FLAG_MUXCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun flag
DMAMUX_FLAG_MUXCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun flag
DMAMUX_FLAG_MUXCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun flag
DMAMUX_FLAG_MUXCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun flag
DMAMUX_FLAG_MUXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun flag
DMAMUX_FLAG_MUXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun flag
DMAMUX_FLAG_MUXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun flag
DMAMUX_FLAG_MUXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun flag
DMAMUX_FLAG_GENCH0_TO	DMAMUX request generator channel 0 trigger overrun flag
DMAMUX_FLAG_GENCH1_TO	DMAMUX request generator channel 1 trigger overrun flag
DMAMUX_FLAG_GENCH2_TO	DMAMUX request generator channel 2 trigger overrun flag
DMAMUX_FLAG_GENCH3_TO	DMAMUX request generator channel 3 trigger overrun flag

Enum dmamux_interrupt_flag_enum

Table 3-228. Enum dmamux_interrupt_flag_enum

Member name	Function description
-------------	----------------------

DMAMUX_INT_FLAG_MUXCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_GENCH0_TO	DMAMUX request generator channel 0 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH1_TO	DMAMUX request generator channel 1 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH2_TO	DMAMUX request generator channel 2 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH3_TO	DMAMUX request generator channel 3 trigger overrun interrupt flag

dma_deinit

The description of dma_deinit is shown as below:

Table 3-229. Function dma_deinit

Function name	dma_deinit
Function prototype	void dma_deinit(dma_channel_enum channelx);
Function descriptions	deinitialize DMA a channel registers
Precondition	-
The called functions	-
Input parameter{in}	

channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize DMA channel 0 registers*/
dma_deinit(DMA_CH0);
```

dma_struct_para_init

The description of dma_struct_para_init is shown as below:

Table 3-230. Function dma_struct_para_init

Function name	dma_struct_para_init
Function prototype	void dma_struct_para_init(dma_parameter_struct* init_struct);
Function descriptions	initialize the parameters of DMA struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	the initialization data needed to initialize DMA channel, refer to Table 3-220. Structure dma_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

dma_init

The description of dma_init is shown as below:

Table 3-231. Function dma_init

Function name	dma_init
Function prototype	void dma_init(dma_channel_enum channelx, dma_parameter_struct init_struct);
Function descriptions	initialize DMA channel
Precondition	-

The called functions	-
Input parameter{in}	
channelx	DMA channel
DMA_CHx(x=0..6)	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to Table 3-220. Structure dma_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* DMA channel 0 initialize */
dma_parameter_struct dma_init_struct;

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

```

dma_circulation_enable

The description of dma_circulation_enable is shown as below:

Table 3-232. Function dma_circulation_enable

Function name	dma_circulation_enable
Function prototype	void dma_circulation_enable(dma_channel_enum channelx);
Function descriptions	enable DMA circulation mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
DMA_CHx(x=0..6)	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable DMA channel 0 circulation mode */
dma_circulation_enable(DMA_CH0);
```

dma_circulation_disable

The description of dma_circulation_disable is shown as below:

Table 3-233. Function dma_circulation_disable

Function name	dma_circulation_disable
Function prototype	void dma_circulation_disable(dma_channel_enum channelx);
Function descriptions	disable DMA circulation mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA channel 0 circulation mode */
dma_circulation_disable(DMA_CH0);
```

dma_memory_to_memory_enable

The description of dma_memory_to_memory_enable is shown as below:

Table 3-234. Function dma_memory_to_memory_enable

Function name	dma_memory_to_memory_enable
Function prototype	void dma_memory_to_memory_enable(dma_channel_enum channelx);
Function descriptions	enable memory to memory mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable DMA channel 0 memory to memory mode */
dma_memory_to_memory_enable(DMA_CH0);
```

dma_memory_to_memory_disable

The description of dma_memory_to_memory_disable is shown as below:

Table 3-235. Function dma_memory_to_memory_disable

Function name	dma_memory_to_memory_disable
Function prototype	void dma_memory_to_memory_disable(dma_channel_enum channelx);
Function descriptions	disable memory to memory mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*disable DMA channel0 memory to memory mode */
dma_memory_to_memory_disable(DMA_CH0);
```

dma_channel_enable

The description of dma_channel_enable is shown as below:

Table 3-236. Function dma_channel_enable

Function name	dma_channel_enable
Function prototype	void dma_channel_enable(dma_channel_enum channelx);
Function descriptions	enable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable DMA channel 0 */
dma_channel_enable(DMA_CH0);
```

dma_channel_disable

The description of dma_channel_disable is shown as below:

Table 3-237. Function dma_channel_disable

Function name	dma_channel_disable
Function prototype	void dma_channel_disable(dma_channel_enum channelx);
Function descriptions	disable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA channel 0 */
dma_channel_disable(DMA_CH0);
```

dma_periph_address_config

The description of dma_periph_address_config is shown as below:

Table 3-238. Function dma_periph_address_config

Function name	dma_periph_address_config
Function prototype	void dma_periph_address_config(dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA peripheral base address
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
address	peripheral base address

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel 0 periph address */
#define BANK0_WRITE_START_ADDR          ((uint32_t)0x08004000)
dma_periph_address_config(DMA_CH0, BANK0_WRITE_START_ADDR);
```

dma_memory_address_config

The description of dma_memory_address_config is shown as below:

Table 3-239. Function dma_memory_address_config

Function name	dma_memory_address_config
Function prototype	void dma_memory_address_config(dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA memory base address
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
address	memory base address, 0 – 0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel 0 memory address */
uint8_t g_destbuf[TRANSFER_NUM];
dma_memory_address_config(DMA_CH0, (uint32_t) g_destbuf);
```

dma_transfer_number_config

The description of dma_transfer_number_config is shown as below:

Table 3-240. Function dma_transfer_number_config

Function name	dma_transfer_number_config
Function prototype	void dma_transfer_number_config(dma_channel_enum channelx, uint32_t number);
Function descriptions	set the number of remaining data to be transferred by the DMA

Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
number	data transfer number(0x0-0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel 0 transfer number */
#define TRANSFER_NUM          0x400
dma_transfer_number_config(DMA_CH0, TRANSFER_NUM);
```

dma_transfer_number_get

The description of dma_transfer_number_get is shown as below:

Table 3-241. Function dma_transfer_number_get

Function name	dma_transfer_number_get
Function prototype	uint32_t dma_transfer_number_get(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}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
uint32_t	DMA data transmission remaining quantity (0x0-0xFFFF)

Example:

```
/* get DMA channel 0 transfer number */
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 3-242. Function dma_priority_config

Function name	dma_priority_config
Function prototype	void dma_priority_config(dma_channel_enum channelx, uint32_t priority);
Function descriptions	configure priority level of DMA channel
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. 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 DMA channel 0 priority */
dma_priority_config(DMA_CH0, DMA_PRIORITY_ULTRA_HIGH);
```

dma_memory_width_config

The description of dma_memory_width_config is shown as below:

Table 3-243. Function dma_memory_width_config

Function name	dma_memory_width_config
Function prototype	void dma_memory_width_config(dma_channel_enum channelx, uint32_t mwidth);
Function descriptions	configure transfer data size of memory
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
mwidth	transfer data width of memory
<i>DMA_MEMORY_WIDTH</i>	transfer data width of memory is 8-bit

<i>H_8BIT</i>	
<i>DMA_MEMORY_WIDT</i> <i>H_16BIT</i>	transfer data width of memory is 16-bit
<i>DMA_MEMORY_WIDT</i> <i>H_32BIT</i>	transfer data width of memory is 32-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel 0 memory width */
dma_memory_width_config(DMA_CH0, DMA_MEMORY_WIDTH_8BIT);
```

dma_periph_width_config

The description of dma_periph_width_config is shown as below:

Table 3-244. Function dma_periph_width_config

Function name	dma_periph_width_config
Function prototype	void dma_periph_width_config(dma_channel_enum channelx, uint32_t pwidth);
Function descriptions	configure transfer data width of peripheral
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx</i> (x=0..6)	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
pwidth	transfer data width of peripheral
<i>DMA_PERIPHERAL_W</i> <i>IDTH_8BIT</i>	transfer data width of peripheral is 8-bit
<i>DMA_PERIPHERAL_W</i> <i>IDTH_16BIT</i>	transfer data width of peripheral is 16-bit
<i>DMA_PERIPHERAL_W</i> <i>IDTH_32BIT</i>	transfer data width of peripheral is 32-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel 0 periph width */
dma_periph_width_config(DMA_CH0, DMA_PERIPHERAL_WIDTH_8BIT);
```


dma_memory_increase_enable

The description of dma_memory_increase_enable is shown as below:

Table 3-245. Function dma_memory_increase_enable

Function name	dma_memory_increase_enable
Function prototype	void dma_memory_increase_enable(dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of memory
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA channel 0 memory increase */
dma_memory_increase_enable(DMA_CH0);
```

dma_memory_increase_disable

The description of dma_memory_increase_disable is shown as below:

Table 3-246. Function dma_memory_increase_disable

Function name	dma_memory_increase_disable
Function prototype	void dma_memory_increase_disable(dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of memory
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA channel 0 memory increase */
dma_memory_increase_disable(DMA_CH0);
```

dma_periph_increase_enable

The description of dma_periph_increase_enable is shown as below:

Table 3-247. Function dma_periph_increase_enable

Function name	dma_periph_increase_enable
Function prototype	void dma_periph_increase_enable(dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of peripheral
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable next address increasement algorithm of DMA channel 0 */
dma_periph_increase_enable(DMA_CH0);
```

dma_periph_increase_disable

The description of dma_periph_increase_disable is shown as below:

Table 3-248. Function dma_periph_increase_disable

Function name	dma_periph_increase_disable
Function prototype	void dma_periph_increase_disable(dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of peripheral
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable next address increasement algorithm of DMA channel 0 */
dma_periph_increase_disable(DMA_CH0);
```

dma_transfer_direction_config

The description of dma_transfer_direction_config is shown as below:

Table 3-249. Function dma_transfer_direction_config

Function name	dma_transfer_direction_config
Function prototype	void dma_transfer_direction_config(dma_channel_enum channelx, uint32_t direction);
Function descriptions	configure the direction of data transfer on the channel
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. 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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA channel0 transfer direction */
dma_transfer_direction_config(DMA_CH0, DMA_PERIPHERAL_TO_MEMORY);
```

dma_flag_get

The description of dma_flag_get is shown as below:

Table 3-250. Function dma_flag_get

Function name	dma_flag_get
Function prototype	FlagStatus dma_flag_get(dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
flag	specify get which flag

<i>DMA_FLAG_G</i>	global interrupt flag of channel
<i>DMA_FLAG_FTF</i>	full transfer finish flag of channel
<i>DMA_FLAG_HTF</i>	half transfer finish flag of channel
<i>DMA_FLAG_ERR</i>	error flag of channel
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMA channel 0 flag */
FlagStatus flag = RESET;
flag = dma_flag_get(DMA_CH0, DMA_FLAG_FTF);
```

dma_flag_clear

The description of dma_flag_clear is shown as below:

Table 3-251. Function dma_flag_clear

Function name	dma_flag_clear
Function prototype	void dma_flag_clear(dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear DMA a channel flag
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
flag	specify get which flag
<i>DMA_FLAG_G</i>	global interrupt flag of channel
<i>DMA_FLAG_FTF</i>	full transfer finish flag of channel
<i>DMA_FLAG_HTF</i>	half transfer finish flag of channel
<i>DMA_FLAG_ERR</i>	error flag of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMA channel 0 flag */
dma_flag_clear(DMA_CH0, DMA_FLAG_FTF);
```

dma_interrupt_enable

The description of dma_interrupt_enable is shown as below:

Table 3-252. Function dma_interrupt_enable

Function name	dma_interrupt_enable
Function prototype	void dma_interrupt_enable(dma_channel_enum channelx, uint32_t source);
Function descriptions	enable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
source	DMA interrupt source
<i>DMA_INT_FTF</i>	full transfer finish interrupt of channel
<i>DMA_INT_HTF</i>	half transfer finish interrupt of channel
<i>DMA_INT_ERR</i>	error interrupt of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA channel 0 interrupt */
dma_interrupt_enable(DMA_CH0, DMA_INT_FTF);
```

dma_interrupt_disable

The description of dma_interrupt_disable is shown as below:

Table 3-253. Function dma_interrupt_disable

Function name	dma_interrupt_disable
Function prototype	void dma_interrupt_disable(dma_channel_enum channelx, uint32_t source);
Function descriptions	disable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
source	DMA interrupt source
<i>DMA_INT_FTF</i>	full transfer finish interrupt of channel
<i>DMA_INT_HTF</i>	half transfer finish interrupt of channel

<i>DMA_INT_ERR</i>	error interrupt of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA channel 0 interrupt */
dma_interrupt_disable(DMA_CH0, DMA_INT_FTF);
```

dma_interrupt_flag_get

The description of dma_interrupt_flag_get is shown as below:

Table 3-254. Function dma_interrupt_flag_get

Function name	dma_interrupt_flag_get
Function prototype	FlagStatus dma_interrupt_flag_get(dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag and interrupt enable bit is set or not
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
flag	specify get which flag
<i>DMA_INT_FLAG_FTF</i>	full transfer finish interrupt flag of channel
<i>DMA_INT_FLAG_HTF</i>	half transfer finish interrupt flag of channel
<i>DMA_INT_FLAG_ERR</i>	error interrupt flag of channel
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMA channel 3 interrupt flag */
if(dma_interrupt_flag_get(DMA_CH3, DMA_INT_FLAG_FTF)){
    dma_interrupt_flag_clear(DMA_CH3, DMA_INT_FLAG_G);
}
```

dma_interrupt_flag_clear

The description of dma_interrupt_flag_clear is shown as below:

Table 3-255. Function dma_interrupt_flag_clear

Function name	dma_interrupt_flag_clear
Function prototype	void dma_interrupt_flag_clear(dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear DMA a channel flag
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMA channel
<i>DMA_CHx(x=0..6)</i>	DMA channel selection, refer to Table 3-223. Enum dma_channel_enum
Input parameter{in}	
flag	specify get which flag
<i>DMA_INT_FLAG_G</i>	global interrupt flag of channel
<i>DMA_INT_FLAG_FTF</i>	full transfer finish interrupt flag of channel
<i>DMA_INT_FLAG_HTF</i>	half transfer finish interrupt flag of channel
<i>DMA_INT_FLAG_ERR</i>	error interrupt flag of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* clear DMA channel 3 interrupt flag */
if(dma_interrupt_flag_get(DMA_CH3, DMA_INT_FLAG_FTF)){
    dma_interrupt_flag_clear(DMA_CH3, DMA_INT_FLAG_G);
}

```

dmamux_sync_struct_para_init

The description of dmamux_sync_struct_para_init is shown as below:

Table 3-256. Function dmamux_sync_struct_para_init

Function name	dmamux_sync_struct_para_init
Function prototype	void dmamux_sync_struct_para_init(dmamux_sync_parameter_struct *init_struct);
Function descriptions	initialize the parameters of DMAMUX synchronization mode structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
init_struct	the initialization data needed to initialize DMAMUX request multiplexer channel synchronization mode, refer to Table 3-221. Structure

	<u>dmamux_sync_parameter_struct</u>
Return value	
-	-

Example:

```
/* initialize DMAMUX synchronization mode structure */
dmamux_sync_parameter_struct dmamux_sync_init_struct;
dmamux_sync_struct_para_init(&dmamux_sync_init_struct);
```

dmamux_synchronization_init

The description of dmamux_synchronization_init is shown as below:

Table 3-257. Function dmamux_synchronization_init

Function name	dmamux_synchronization_init
Function prototype	void dmamux_synchronization_init(dmamux_multiplexer_channel_enum channelx, dmamux_sync_parameter_struct *init_struct);
Function descriptions	initialize DMAMUX request multiplexer channel synchronization mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..6)	DMAMUX channel selection, refer to <u>Table 3-224. Enum dmamux_multiplexer_channel_enum</u>
Input parameter{in}	
init_struct	the initialization data needed to initialize DMAMUX request multiplexer channel synchronization mode, refer to <u>Table 3-221. Structure dmamux_sync_parameter_struct</u>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize DMAMUX synchronization mode structure */
dmamux_sync_parameter_struct dmamux_sync_init_struct;
dmamux_sync_struct_para_init(&dmamux_sync_init_struct);
/* initialize DMA request multiplexer channel 0 with synchronization mode */
dmamux_sync_init_struct.sync_id      = DMAMUX_SYNC_EXTI0;
dmamux_sync_init_struct.sync_polarity = DMAMUX_SYNC_RISING;
dmamux_sync_init_struct.request_number = 4;
dmamux_synchronization_init(DMAMUX_MUXCH0, &dmamux_sync_init_struct);
```


dmamux_synchronization_enable

The description of dmamux_synchronization_enable is shown as below:

Table 3-258. Function dmamux_synchronization_enable

Function name	dmamux_synchronization_enable
Function prototype	void dmamux_synchronization_enable(dmamux_multiplexer_channel_enum channelx);
Function descriptions	enable synchronization mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..6)	DMAMUX channel selection, refer to Table 3-224. Enum dmamux_multiplexer_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable synchronization mode */
dmamux_synchronization_enable(DMAMUX_MUXCH0);
```

dmamux_synchronization_disable

The description of dmamux_synchronization_disable is shown as below:

Table 3-259. Function dmamux_synchronization_disable

Function name	dmamux_synchronization_disable
Function prototype	void dmamux_synchronization_disable(dmamux_multiplexer_channel_enum channelx);
Function descriptions	disable synchronization mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..6)	DMAMUX channel selection, refer to Table 3-224. Enum dmamux_multiplexer_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable synchronization mode */
dmamux_synchronization_disable(DMAMUX_MUXCH0);
```

dmamux_event_generation_enable

The description of dmamux_event_generation_enable is shown as below:

Table 3-260. Function dmamux_event_generation_enable

Function name	dmamux_event_generation_enable
Function prototype	void dmamux_event_generation_enable(dmamux_multiplexer_channel_enum channelx);
Function descriptions	enable event generation
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..6)	DMAMUX channel selection, refer to Table 3-224. Enum dmamux_multiplexer_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable event generation */
dmamux_event_generation_enable(DMAMUX_MUXCH0);
```

dmamux_event_generation_disable

The description of dmamux_event_generation_disable is shown as below:

Table 3-261. Function dmamux_event_generation_disable

Function name	dmamux_event_generation_disable
Function prototype	void dmamux_event_generation_disable(dmamux_multiplexer_channel_enum channelx);
Function descriptions	disable event generation
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=	DMAMUX channel selection, refer to Table 3-224. Enum

0..6)	dmamux_muxchannel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable event generation */
dmamux_event_generation_disable(DMAMUX_MUXCH0);
```

dmamux_gen_struct_para_init

The description of dmamux_gen_struct_para_init is shown as below:

Table 3-262. Function dmamux_gen_struct_para_init

Function name	dmamux_gen_struct_para_init
Function prototype	void dmamux_gen_struct_para_init(dmamux_gen_parameter_struct *init_struct);
Function descriptions	initialize the parameters of DMAMUX request generator structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
init_struct	the initialization data needed to initialize DMAMUX request generator channel, refer to Table 3-222. Structure dmamux_gen_parameter_struct
Return value	
-	-

Example:

```
/* initialize DMA request generator structure */
dmamux_gen_parameter_struct dmamux_gen_init_struct;
dmamux_gen_struct_para_init(&dmamux_gen_init_struct);
```

dmamux_request_generator_init

The description of dmamux_request_generator_init is shown as below:

Table 3-263. Function dmamux_request_generator_init

Function name	dmamux_request_generator_init
Function prototype	void dmamux_request_generator_init(dmamux_generator_channel_enum channelx, dmamux_gen_parameter_struct *init_struct);
Function descriptions	initialize DMAMUX request generator channel
Precondition	-

The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
<i>DMAMUX_GENCHx(x=0..3)</i>	DMAMUX generation channel selection, refer to Table 3-225. Enum dmamux_generator_channel_enum
Input parameter{in}	
init_struct	the initialization data needed to initialize DMAMUX request generator channel, refer to Table 3-222. Structure dmamux_gen_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize DMA request generator channel 0 */
dmamux_gen_parameter_struct  dmamux_gen_init_struct;
dmamux_gen_struct_para_init(&dmamux_gen_init_struct);
dmamux_gen_init_struct.trigger_id      = DMAMUX_TRIGGER_EXTI13;
dmamux_gen_init_struct.trigger_polarity = DMAMUX_GEN_RISING;
dmamux_gen_init_struct.request_number = 1;
dmamux_request_generator_init(DMAMUX_GENCH0, &dmamux_gen_init_struct);

```

dmamux_request_generator_channel_enable

The description of dmamux_request_generator_channel_enable is shown as below:

Table 3-264. Function dmamux_request_generator_channel_enable

Function name	dmamux_request_generator_channel_enable
Function prototype	void dmamux_request_generator_channel_enable(dmamux_generator_channel_enum channelx);
Function descriptions	enable DMAMUX request generator channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
<i>DMAMUX_GENCHx(x=0..3)</i>	DMAMUX generation channel selection, refer to Table 3-225. Enum dmamux_generator_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMAMUX request generator channel */
```

```
dmamux_request_generator_channel_enable(DMAMUX_GENCH0);
```

dmamux_request_generator_channel_disable

The description of dmamux_request_generator_channel_disable is shown as below:

Table 3-265. Function dmamux_request_generator_channel_disable

Function name	dmamux_request_generator_channel_disable
Function prototype	void dmamux_request_generator_channel_disable(dmamux_generator_channel_enum channelx);
Function descriptions	disable DMAMUX request generator channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
DMAMUX_GENCHx(x=0..3)	DMAMUX generation channel selection, refer to Table 3-225. Enum dmamux_generator_channel_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMAMUX request generator channel */
```

```
dmamux_request_generator_channel_disable(DMAMUX_GENCH0);
```

dmamux_synchronization_polarity_config

The description of dmamux_synchronization_polarity_config is shown as below:

Table 3-266. Function dmamux_synchronization_polarity_config

Function name	dmamux_synchronization_polarity_config
Function prototype	void dmamux_synchronization_polarity_config(dmamux_multiplexer_channel_enum channelx, uint32_t polarity);
Function descriptions	configure synchronization input polarity
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..6)	DMAMUX channel selection, refer to Table 3-224. Enum dmamux_multiplexer_channel_enum
Input parameter{in}	

polarity	synchronization input polarity
<i>DMAMUX_SYNC_NO_EVENT</i>	no event detection
<i>DMAMUX_SYNC_RISING</i>	rising edge
<i>DMAMUX_SYNC_FALLING</i>	falling edge
<i>DMAMUX_SYNC_RISING_FALLING</i>	rising and falling edges
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure synchronization input polarity */
dmamux_synchronization_polarity_config(DMAMUX_MUXCH0, DMAMUX_SYNC_RISING);
```

dmamux_request_forward_number_config

The description of dmamux_request_forward_number_config is shown as below:

Table 3-267. Function dmamux_request_forward_number_config

Function name	dmamux_request_forward_number_config
Function prototype	void dmamux_request_forward_number_config(dmamux_mux_channel_enum channelx, uint32_t number);
Function descriptions	configure number of DMA requests to forward
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
<i>DMAMUX_MUXCHx(x=0..6)</i>	DMAMUX channel selection, refer to Table 3-224. Enum dmamux_mux_channel_enum
Input parameter{in}	
number	DMA requests number to forward (1 - 32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure number of DMA requests to forward */
dmamux_request_forward_number_config(DMAMUX_MUXCH0, 4);
```

dmamux_sync_id_config

The description of dmamux_sync_id_config is shown as below:

Table 3-268. Function dmamux_sync_id_config

Function name	dmamux_sync_id_config
Function prototype	void dmamux_sync_id_config(dmamux_multiplexer_channel_enum channelx, uint32_t id);
Function descriptions	configure synchronization input identification
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..6)	DMAMUX channel selection, refer to Table 3-224. Enum dmamux_multiplexer_channel_enum
Input parameter{in}	
id	synchronization input identification
DMAMUX_SYNC_EXTI0	synchronization input is EXTI0
DMAMUX_SYNC_EXTI1	synchronization input is EXTI1
DMAMUX_SYNC_EXTI2	synchronization input is EXTI2
DMAMUX_SYNC_EXTI3	synchronization input is EXTI3
DMAMUX_SYNC_EXTI4	synchronization input is EXTI4
DMAMUX_SYNC_EXTI5	synchronization input is EXTI5
DMAMUX_SYNC_EXTI6	synchronization input is EXTI6
DMAMUX_SYNC_EXTI7	synchronization input is EXTI7
DMAMUX_SYNC_EXTI8	synchronization input is EXTI8
DMAMUX_SYNC_EXTI9	synchronization input is EXTI9
DMAMUX_SYNC_EXTI10	synchronization input is EXTI10
DMAMUX_SYNC_EXTI11	synchronization input is EXTI11
DMAMUX_SYNC_EXTI12	synchronization input is EXTI12

<i>DMAMUX_SYNC_EXTI13</i>	synchronization input is EXTI13
<i>DMAMUX_SYNC_EXTI14</i>	synchronization input is EXTI14
<i>DMAMUX_SYNC_EXTI15</i>	synchronization input is EXTI15
<i>DMAMUX_SYNC_EVTX_OUT0</i>	synchronization input is Evt_out0
<i>DMAMUX_SYNC_EVTX_OUT1</i>	synchronization input is Evt_out1
<i>DMAMUX_SYNC_EVTX_OUT2</i>	synchronization input is Evt_out2
<i>DMAMUX_SYNC_EVTX_OUT3</i>	synchronization input is Evt_out3
<i>DMAMUX_SYNC_TIMER20_CH0_O</i>	synchronization input is TIMER20_CH0_O
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure synchronization input identification */
dmamux_sync_id_config(DMAMUX_MUXCH0, DMAMUX_SYNC_EXTI0);
```

dmamux_request_id_config

The description of dmamux_request_id_config is shown as below:

Table 3-269. Function dmamux_request_id_config

Function name	dmamux_request_id_config
Function prototype	void dmamux_request_id_config(dmamux_multiplexer_channel_enum channelx, uint32_t id);
Function descriptions	configure multiplexer input identification
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
<i>DMAMUX_MUXCHx</i> (x=0..6)	DMAMUX channel selection, refer to Table 3-224. Enum dmamux_multiplexer_channel_enum
Input parameter{in}	
id	input DMA request identification
<i>DMA_REQUEST_M2M</i>	memory to memory transfer
<i>DMA_REQUEST_GEN</i>	DMAMUX request generator 0

<i>ERATOR0</i>	
<i>DMA_REQUEST_GEN ERATOR1</i>	DMAMUX request generator 1
<i>DMA_REQUEST_GEN ERATOR2</i>	DMAMUX request generator 2
<i>DMA_REQUEST_GEN ERATOR3</i>	DMAMUX request generator 3
<i>DMA_REQUEST_ADC</i>	DMAMUX ADC request
<i>DMA_REQUEST_DAC _CH0</i>	DMAMUX DAC CH0 request
<i>DMA_REQUEST_I2C1 _RX</i>	DMAMUX I2C1 RX request
<i>DMA_REQUEST_I2C1 _TX</i>	DMAMUX I2C1 TX request
<i>DMA_REQUEST_I2C0 _RX</i>	DMAMUX I2C0 RX request
<i>DMA_REQUEST_I2C0 _TX</i>	DMAMUX I2C0 TX request
<i>DMA_REQUESTR_SST AT0</i>	DMAMUX SSTAT0 request
<i>DMA_REQUESTR_SST AT1</i>	DMAMUX SSTAT1 request
<i>DMA_REQUESTR_SST AT2</i>	DMAMUX SSTAT2 request
<i>DMA_REQUESTR_SST AT3</i>	DMAMUX SSTAT3 request
<i>DMA_REQUEST_SPI0 _RX</i>	DMAMUX SPI0 RX request
<i>DMA_REQUEST_SPI0 _TX</i>	DMAMUX SPI0 TX request
<i>DMA_REQUEST_SPI1 _RX</i>	DMAMUX SPI1 RX request
<i>DMA_REQUEST_SPI1 _TX</i>	DMAMUX SPI1 TX request
<i>DMA_REQUEST_TIME R0_CH0</i>	DMAMUX TIMER0 CH0 request
<i>DMA_REQUEST_TIME R0_CH1</i>	DMAMUX TIMER0 CH1 request
<i>DMA_REQUEST_TIME R0_CH2</i>	DMAMUX TIMER0 CH2 request
<i>DMA_REQUEST_TIME R0_CH3</i>	DMAMUX TIMER0 CH3 request
<i>DMA_REQUEST_TIME</i>	DMAMUX TIMER0 TI request

<i>R0_TI</i>	
<i>DMA_REQUEST_TIME</i> <i>R0_UP</i>	DMAMUX TIMER0 UP request
<i>DMA_REQUEST_TIME</i> <i>R0_CO</i>	DMAMUX TIMER0 CO request
<i>DMA_REQUEST_TIME</i> <i>R0_MCH0</i>	DMAMUX TIMER0 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R0_MCH1</i>	DMAMUX TIMER0 MCH1 request
<i>DMA_REQUEST_TIME</i> <i>R0_MCH2</i>	DMAMUX TIMER0 MCH2 request
<i>DMA_REQUEST_TIME</i> <i>R0_MCH3</i>	DMAMUX TIMER0 MCH3 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH0</i>	DMAMUX TIMER1 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH1</i>	DMAMUX TIMER1 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH2</i>	DMAMUX TIMER1 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH3</i>	DMAMUX TIMER1 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R1_TI</i>	DMAMUX TIMER1 TI request
<i>DMA_REQUEST_TIME</i> <i>R1_UP</i>	DMAMUX TIMER1 UP request
<i>DMA_REQUEST_TIME</i> <i>R7_CH0</i>	DMAMUX TIMER7 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R7_CH1</i>	DMAMUX TIMER7 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R7_CH2</i>	DMAMUX TIMER7 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R7_CH3</i>	DMAMUX TIMER7 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R7_TI</i>	DMAMUX TIMER7 TI request
<i>DMA_REQUEST_TIME</i> <i>R7_UP</i>	DMAMUX TIMER7 UP request
<i>DMA_REQUEST_TIME</i> <i>R7_CO</i>	DMAMUX TIMER7 CO request
<i>DMA_REQUEST_TIME</i> <i>R7_MCH0</i>	DMAMUX TIMER7 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R7_MCH1</i>	DMAMUX TIMER7 MCH1 request

<i>DMA_REQUEST_TIME</i> <i>R7_MCH2</i>	DMAMUX TIMER7 MCH2 request
<i>DMA_REQUEST_TIME</i> <i>R7_MCH3</i>	DMAMUX TIMER7 MCH3 request
<i>DMA_REQUEST_CAN</i> <i>1</i>	DMAMUX CAN1 request
<i>DMA_REQUEST_CAN</i> <i>0</i>	DMAMUX CAN0 request
<i>DMA_REQUEST_USA</i> <i>RT0_RX</i>	DMAMUX USART0 RX request
<i>DMA_REQUEST_USA</i> <i>RT0_TX</i>	DMAMUX USART0 TX request
<i>DMA_REQUEST_USA</i> <i>RT1_RX</i>	DMAMUX USART1 RX request
<i>DMA_REQUEST_USA</i> <i>RT1_TX</i>	DMAMUX USART1 TX request
<i>DMA_REQUEST_USA</i> <i>RT2_RX</i>	DMAMUX USART2 RX request
<i>DMA_REQUEST_USA</i> <i>RT2_TX</i>	DMAMUX USART2 TX request
<i>DMA_REQUEST_TIME</i> <i>R5_UP</i>	DMAMUX TIMER5 UP request
<i>DMA_REQUEST_TIME</i> <i>R6_UP</i>	DMAMUX TIMER6 UP request
<i>DMA_REQUEST_TIME</i> <i>R19_CH0</i>	DMAMUX TIMER19 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R19_CH1</i>	DMAMUX TIMER19 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R19_CH2</i>	DMAMUX TIMER19 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R19_CH3</i>	DMAMUX TIMER19 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R19_TI</i>	DMAMUX TIMER19 TI request
<i>DMA_REQUEST_TIME</i> <i>R19_UP</i>	DMAMUX TIMER19 UP request
<i>DMA_REQUEST_TIME</i> <i>R19_CO</i>	DMAMUX TIMER19 CO request
<i>DMA_REQUEST_TIME</i> <i>R19_MCH0</i>	DMAMUX TIMER19 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R19_MCH1</i>	DMAMUX TIMER19 MCH1 request
<i>DMA_REQUEST_TIME</i>	DMAMUX TIMER191 MCH2 request

<i>R19_MCH2</i>	
<i>DMA_REQUEST_TIME</i> <i>R19_MCH3</i>	DMAMUX TIMER19 MCH3 request
<i>DMA_REQUEST_TIME</i> <i>R20_CH0</i>	DMAMUX TIMER20 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R20_CH1</i>	DMAMUX TIMER20 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R20_CH2</i>	DMAMUX TIMER20 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R20_CH3</i>	DMAMUX TIMER20 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R20_TI</i>	DMAMUX TIMER20 TI request
<i>DMA_REQUEST_TIME</i> <i>R20_UP</i>	DMAMUX TIMER20 UP request
<i>DMA_REQUEST_TIME</i> <i>R20_CO</i>	DMAMUX TIMER20 CO request
<i>DMA_REQUEST_TIME</i> <i>R20_MCH0</i>	DMAMUX TIMER20 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R20_MCH1</i>	DMAMUX TIMER20 MCH1 request
<i>DMA_REQUEST_TIME</i> <i>R20_MCH2</i>	DMAMUX TIMER20 MCH2 request
<i>DMA_REQUEST_TIME</i> <i>R20_MCH3</i>	DMAMUX TIMER20 MCH3 request
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure multiplexer input identification */
dmamux_request_id_config(DMAMUX_MUXCH0, DMA_REQUEST_GENERATOR0);
```

dmamux_trigger_polarity_config

The description of dmamux_trigger_polarity_config is shown as below:

Table 3-270. Function dmamux_trigger_polarity_config

Function name	dmamux_trigger_polarity_config
Function prototype	void dmamux_trigger_polarity_config(dmamux_generator_channel_enum channelx, uint32_t polarity);
Function descriptions	configure trigger input polarity
Precondition	-

The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
DMAMUX_GENCHx(x=0..3)	DMAMUX generation channel selection, refer to Table 3-225. Enum dmamux_generator_channel_enum
Input parameter{in}	
polarity	trigger input polarity
DMAMUX_GEN_NO_EVENT	no event detection
DMAMUX_GEN_RISING	rising edge
DMAMUX_GEN_FALLING	falling edge
DMAMUX_GEN_RISING_FALLING	rising and falling edges
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure trigger input polarity */
dmamux_trigger_polarity_config(DMAMUX_GENCH0, DMAMUX_GEN_RISING);
```

dmamux_request_generate_number_config

The description of dmamux_request_generate_number_config is shown as below:

Table 3-271. Function dmamux_request_generate_number_config

Function name	dmamux_request_generate_number_config
Function prototype	void dmamux_request_generate_number_config(dmamux_generator_channel_enum channelx, uint32_t number);
Function descriptions	configure number of DMA requests to be generated
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
DMAMUX_GENCHx(x=0..3)	DMAMUX generation channel selection, refer to Table 3-225. Enum dmamux_generator_channel_enum
Input parameter{in}	
number	DMA requests number to be generated (1 - 32)
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* configure number of DMA requests to be generated */
dmamux_request_generate_number_config(DMAMUX_GENCH0, 1);
```

dmamux_trigger_id_config

The description of dmamux_trigger_id_config is shown as below:

Table 3-272. Function dmamux_trigger_id_config

Function name	dmamux_trigger_id_config
Function prototype	void dmamux_trigger_id_config(dmamux_generator_channel_enum channelx, uint32_t id);
Function descriptions	configure trigger input identification
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
DMAMUX_GENCHx(x=0..3)	DMAMUX generation channel selection, refer to Table 3-225. Enum dmamux_generator_channel_enum
Input parameter{in}	
id	trigger input identification
DMAMUX_TRIGGER_EXTI0	trigger input is EXTI0
DMAMUX_TRIGGER_EXTI1	trigger input is EXTI1
DMAMUX_TRIGGER_EXTI2	trigger input is EXTI2
DMAMUX_TRIGGER_EXTI3	trigger input is EXTI3
DMAMUX_TRIGGER_EXTI4	trigger input is EXTI4
DMAMUX_TRIGGER_EXTI5	trigger input is EXTI5
DMAMUX_TRIGGER_EXTI6	trigger input is EXTI6
DMAMUX_TRIGGER_EXTI7	trigger input is EXTI7
DMAMUX_TRIGGER_EXTI8	trigger input is EXTI8
DMAMUX_TRIGGER_EXTI9	trigger input is EXTI9

DMAMUX_TRIGGER_EXTI10	trigger input is EXTI10
DMAMUX_TRIGGER_EXTI11	trigger input is EXTI11
DMAMUX_TRIGGER_EXTI12	trigger input is EXTI12
DMAMUX_TRIGGER_EXTI13	trigger input is EXTI13
DMAMUX_TRIGGER_EXTI14	trigger input is EXTI14
DMAMUX_TRIGGER_EXTI15	trigger input is EXTI15
DMAMUX_TRIGGER_EVT_OUT0	trigger input is Evt_out0
DMAMUX_TRIGGER_EVT_OUT1	trigger input is Evt_out1
DMAMUX_TRIGGER_EVT_OUT2	trigger input is Evt_out2
DMAMUX_TRIGGER_EVT_OUT3	trigger input is Evt_out3
DMAMUX_TRIGGER_TIMER20_CH0_O	trigger input is TIMER20_CH0_O
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure trigger input identification */
dmamux_trigger_id_config(DMAMUX_GENCH0, DMAMUX_TRIGGER_EXTI13);
```

dmamux_flag_get

The description of dmamux_flag_get is shown as below:

Table 3-273. Function dmamux_flag_get

Function name	dmamux_flag_get
Function prototype	FlagStatus dmamux_flag_get(dmamux_flag_enum flag);
Function descriptions	get DMAMUX flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag type, refer to Table 3-227. Enum dmamux_flag_enum
DMAMUX_FLAG_MUX	DMAMUX request multiplexer channel 0 synchronization overrun flag

<i>CH0_SO</i>	
<i>DMAMUX_FLAG_MUX</i> <i>CH1_SO</i>	DMAMUX request multiplexer channel 1 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH2_SO</i>	DMAMUX request multiplexer channel 2 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH3_SO</i>	DMAMUX request multiplexer channel 3 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH4_SO</i>	DMAMUX request multiplexer channel 4 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH5_SO</i>	DMAMUX request multiplexer channel 5 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH6_SO</i>	DMAMUX request multiplexer channel 6 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH7_SO</i>	DMAMUX request multiplexer channel 7 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH8_SO</i>	DMAMUX request multiplexer channel 8 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH9_SO</i>	DMAMUX request multiplexer channel 9 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH10_SO</i>	DMAMUX request multiplexer channel 10 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH11_SO</i>	DMAMUX request multiplexer channel 11 synchronization overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH0_TO</i>	DMAMUX request generator channel 0 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH1_TO</i>	DMAMUX request generator channel 1 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH2_TO</i>	DMAMUX request generator channel 2 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH3_TO</i>	DMAMUX request generator channel 3 trigger overrun flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus flag = RESET;
/* get DMAMUX flag */
flag = dmamux_flag_get(DMAMUX_FLAG_GENCH0_TO);
```


dmamux_flag_clear

The description of dmamux_flag_clear is shown as below:

Table 3-274. Function dmamux_flag_clear

Function name	dmamux_flag_clear
Function prototype	void dmamux_flag_clear(dmamux_flag_enum flag);
Function descriptions	clear DMAMUX flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag type, refer to Table 3-227. Enum dmamux_flag_enum
DMAMUX_FLAG_MUX CH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun flag
DMAMUX_FLAG_MUX CH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun flag
DMAMUX_FLAG_MUX CH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun flag
DMAMUX_FLAG_MUX CH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun flag
DMAMUX_FLAG_MUX CH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun flag
DMAMUX_FLAG_MUX CH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun flag
DMAMUX_FLAG_MUX CH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun flag
DMAMUX_FLAG_MUX CH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun flag
DMAMUX_FLAG_MUX CH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun flag
DMAMUX_FLAG_MUX CH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun flag
DMAMUX_FLAG_MUX CH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun flag
DMAMUX_FLAG_MUX CH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun flag
DMAMUX_FLAG_GEN CH0_TO	DMAMUX request generator channel 0 trigger overrun flag
DMAMUX_FLAG_GEN CH1_TO	DMAMUX request generator channel 1 trigger overrun flag
DMAMUX_FLAG_GEN CH2_TO	DMAMUX request generator channel 2 trigger overrun flag
DMAMUX_FLAG_GEN	DMAMUX request generator channel 3 trigger overrun flag

<i>CH3_TO</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMAMUX flag */
dmamux_flag_clear(DMAMUX_FLAG_GENCH0_TO);
```

dmamux_interrupt_enable

The description of dmamux_interrupt_enable is shown as below:

Table 3-275. Function dmamux_interrupt_enable

Function name	dmamux_interrupt_enable
Function prototype	void dmamux_interrupt_enable(dmamux_interrupt_enum interrupt);
Function descriptions	enable DMAMUX interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify which interrupt to enable
<i>DMAMUX_INT_MUXC H0_SO</i>	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H1_SO</i>	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H2_SO</i>	DMAMUX request multiplexer channel 2 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H3_SO</i>	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H4_SO</i>	DMAMUX request multiplexer channel 4 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H5_SO</i>	DMAMUX request multiplexer channel 5 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H6_SO</i>	DMAMUX request multiplexer channel 6 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H7_SO</i>	DMAMUX request multiplexer channel 7 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H8_SO</i>	DMAMUX request multiplexer channel 8 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H9_SO</i>	DMAMUX request multiplexer channel 9 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H10_SO</i>	DMAMUX request multiplexer channel 10 synchronization overrun interrupt

<i>DMAMUX_INT_MUXC</i> <i>H11_SO</i>	DMAMUX request multiplexer channel 11 synchronization overrun interrupt
<i>DMAMUX_INT_GENC</i> <i>H0_TO</i>	DMAMUX request generator channel 0 trigger overrun interrupt
<i>DMAMUX_INT_GENC</i> <i>H1_TO</i>	DMAMUX request generator channel 1 trigger overrun interrupt
<i>DMAMUX_INT_GENC</i> <i>H2_TO</i>	DMAMUX request generator channel 2 trigger overrun interrupt
<i>DMAMUX_INT_GENC</i> <i>H3_TO</i>	DMAMUX request generator channel 3 trigger overrun interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMAMUX interrupt */
dmamux_interrupt_enable(DMAMUX_INT_MUXCH0_SO);
```

dmamux_interrupt_disable

The description of dmamux_interrupt_disable is shown as below:

Table 3-276. Function dmamux_interrupt_disable

Function name	dmamux_interrupt_disable
Function prototype	void dmamux_interrupt_disable(dmamux_interrupt_enum interrupt);
Function descriptions	disable DMAMUX interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify which interrupt to disable
<i>DMAMUX_INT_MUXC</i> <i>H0_SO</i>	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC</i> <i>H1_SO</i>	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC</i> <i>H2_SO</i>	DMAMUX request multiplexer channel 2 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC</i> <i>H3_SO</i>	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC</i> <i>H4_SO</i>	DMAMUX request multiplexer channel 4 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC</i> <i>H5_SO</i>	DMAMUX request multiplexer channel 5 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC</i> <i>H6_SO</i>	DMAMUX request multiplexer channel 6 synchronization overrun interrupt

<i>H6_SO</i>	
<i>DMAMUX_INT_MUXC H7_SO</i>	DMAMUX request multiplexer channel 7 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H8_SO</i>	DMAMUX request multiplexer channel 8 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H9_SO</i>	DMAMUX request multiplexer channel 9 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H10_SO</i>	DMAMUX request multiplexer channel 10 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H11_SO</i>	DMAMUX request multiplexer channel 11 synchronization overrun interrupt
<i>DMAMUX_INT_GENC H0_TO</i>	DMAMUX request generator channel 0 trigger overrun interrupt
<i>DMAMUX_INT_GENC H1_TO</i>	DMAMUX request generator channel 1 trigger overrun interrupt
<i>DMAMUX_INT_GENC H2_TO</i>	DMAMUX request generator channel 2 trigger overrun interrupt
<i>DMAMUX_INT_GENC H3_TO</i>	DMAMUX request generator channel 3 trigger overrun interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMAMUX interrupt */
dmamux_interrupt_disable(DMAMUX_INT_MUXCH0_SO);
```

dmamux_interrupt_flag_get

The description of dmamux_interrupt_flag_get is shown as below:

Table 3-277. Function dmamux_interrupt_flag_get

Function name	dmamux_interrupt_flag_get
Function prototype	FlagStatus dmamux_interrupt_flag_get(dmamux_interrupt_flag_enum int_flag);
Function descriptions	get DMAMUX interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	flag type, refer to Table 3-228. Enum dmamux_interrupt_flag_enum
<i>DMAMUX_INT_FLAG_ MUXCH0_SO</i>	DMAMUX request multiplexer channel 0 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_ MUXCH1_SO</i>	DMAMUX request multiplexer channel 1 synchronization overrun interrupt flag

<i>MUXCH1_SO</i>	flag
<i>DMAMUX_INT_FLAG_MUXCH2_SO</i>	DMAMUX request multiplexer channel 2 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH3_SO</i>	DMAMUX request multiplexer channel 3 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH4_SO</i>	DMAMUX request multiplexer channel 4 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH5_SO</i>	DMAMUX request multiplexer channel 5 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH6_SO</i>	DMAMUX request multiplexer channel 6 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH7_SO</i>	DMAMUX request multiplexer channel 7 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH8_SO</i>	DMAMUX request multiplexer channel 8 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH9_SO</i>	DMAMUX request multiplexer channel 9 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH10_SO</i>	DMAMUX request multiplexer channel 10 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH11_SO</i>	DMAMUX request multiplexer channel 11 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_GENCH0_TO</i>	DMAMUX request generator channel 0 trigger overrun interrupt flag
<i>DMAMUX_INT_FLAG_GENCH1_TO</i>	DMAMUX request generator channel 1 trigger overrun interrupt flag
<i>DMAMUX_INT_FLAG_GENCH2_TO</i>	DMAMUX request generator channel 2 trigger overrun interrupt flag
<i>DMAMUX_INT_FLAG_GENCH3_TO</i>	DMAMUX request generator channel 3 trigger overrun interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```

/* check DMAMUX interrupt flag */
if(dmamux_interrupt_flag_get(DMAMUX_INT_FLAG_GENCH0_TO)) {
    dmamux_interrupt_flag_clear(DMAMUX_INT_FLAG_GENCH0_TO);
}

```

dmamux_interrupt_flag_clear

The description of dmamux_interrupt_flag_clear is shown as below:

Table 3-278. Function `dmamux_interrupt_flag_clear`

Function name	<code>dmamux_interrupt_flag_clear</code>
Function prototype	<code>FlagStatus dmamux_interrupt_flag_get(dmamux_interrupt_flag_enum int_flag);</code>
Function descriptions	clear DMAMUX interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	flag type, refer to Table 3-228. Enum dmamux_interrupt_flag_enum
<code>DMAMUX_INT_FLAG_MUXCH0_SO</code>	DMAMUX request multiplexer channel 0 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH1_SO</code>	DMAMUX request multiplexer channel 1 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH2_SO</code>	DMAMUX request multiplexer channel 2 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH3_SO</code>	DMAMUX request multiplexer channel 3 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH4_SO</code>	DMAMUX request multiplexer channel 4 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH5_SO</code>	DMAMUX request multiplexer channel 5 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH6_SO</code>	DMAMUX request multiplexer channel 6 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH7_SO</code>	DMAMUX request multiplexer channel 7 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH8_SO</code>	DMAMUX request multiplexer channel 8 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH9_SO</code>	DMAMUX request multiplexer channel 9 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH10_SO</code>	DMAMUX request multiplexer channel 10 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_MUXCH11_SO</code>	DMAMUX request multiplexer channel 11 synchronization overrun interrupt flag
<code>DMAMUX_INT_FLAG_GENCH0_TO</code>	DMAMUX request generator channel 0 trigger overrun interrupt flag
<code>DMAMUX_INT_FLAG_GENCH1_TO</code>	DMAMUX request generator channel 1 trigger overrun interrupt flag
<code>DMAMUX_INT_FLAG_GENCH2_TO</code>	DMAMUX request generator channel 2 trigger overrun interrupt flag
<code>DMAMUX_INT_FLAG_GENCH3_TO</code>	DMAMUX request generator channel 3 trigger overrun interrupt flag
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* check DMAMUX interrupt flag */
if(dmamux_interrupt_flag_get(DMAMUX_INT_FLAG_GENCH0_TO)) {
    dmamux_interrupt_flag_clear(DMAMUX_INT_FLAG_GENCH0_TO);
}
```

3.10. EXTI

EXTI is the interrupt / event controller in the MCU. It contains up to 25 independent edge detectors and generates interrupt requests or events to the processor. The EXTI registers are listed in chapter [3.10.1](#), the EXTI firmware functions are introduced in chapter [3.10.2](#).

3.10.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

Table 3-279. EXTI Registers

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

3.10.2. Descriptions of Peripheral functions

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

Table 3-280. EXTI firmware function

Function name	Function description
exti_deinit	deinitialize the EXTI
exti_init	initialize the EXTI line x
exti_interrupt_enable	enable the interrupts from EXTI line x
exti_interrupt_disable	disable the interrupts from EXTI line x
exti_event_enable	enable the events from EXTI line x
exti_event_disable	disable the events from EXTI line x
exti_software_interrupt_enable	enable the software interrupt event from EXTI line x
exti_software_interrupt_disable	disable the software interrupt event from EXTI line x
exti_flag_get	get EXTI line x interrupt pending flag

Function name	Function description
exti_flag_clear	clear EXTI line x interrupt pending flag
exti_interrupt_flag_get	get EXTI line x interrupt pending flag
exti_interrupt_flag_clear	clear EXTI line x interrupt pending flag

Enum exti_line_enum

Table 3-281. exti_line_enum

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

Enum exti_mode_enum

Table 3-282. exti_mode_enum

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

Enum exti_trig_type_enum

Table 3-283. exti_trig_type_enum

enum 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 edge or falling edge trigger

exti_deinit

The description of exti_deinit is shown as below:

Table 3-284. Function exti_deinit

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

Example:

```
/* deinitialize the EXTI */
exti_deinit();
```

exti_init

The description of exti_init is shown as below:

Table 3-285. Function exti_init

Function name	exti_init
Function prototype	void exti_init(exti_line_enum linex, exti_mode_enum mode, exti_trig_type_enum trig_type);
Function descriptions	initialize the EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-281. exti_line_enum
Input parameter{in}	

mode	EXTI mode, refer to Table 3-282. exti_mode_enum
Input parameter{in}	
trig_type	trigger type, refer to Table 3-283. exti_trig_type_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure EXTI_0 */
```

```
exti_init(EXTI_0, EXTI_INTERRUPT, EXTI_TRIG_BOTH);
```

exti_interrupt_enable

The description of exti_interrupt_enable is shown as below:

Table 3-286. 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 3-281. exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the interrupts from EXTI line 0 */
```

```
exti_interrupt_enable(EXTI_0);
```

exti_interrupt_disable

The description of exti_interrupt_disable is shown as below:

Table 3-287. 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 3-281. exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_interrupt_disable(EXTI_0);
```

exti_event_enable

The description of exti_event_enable is shown as below:

Table 3-288. 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 3-281. exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_event_enable(EXTI_0);
```

exti_event_disable

The description of exti_event_disable is shown as below:

Table 3-289. 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 3-281. exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_event_disable(EXTI_0);
```

exti_software_interrupt_enable

The description of exti_software_interrupt_enable is shown as below:

Table 3-290. Function exti_software_interrupt_enable

Function name	exti_software_interrupt_enable
Function prototype	void exti_software_interrupt_enable(exti_line_enum linex);
Function descriptions	enable the software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-281. exti_line_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_software_interrupt_enable(EXTI_0);
```

exti_software_interrupt_disable

The description of exti_software_interrupt_disable is shown as below:

Table 3-291. Function exti_software_interrupt_disable

Function name	exti_software_interrupt_disable
Function prototype	void exti_software_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-281. exti_line_enum

Output parameter{out}	
-	-
Return value	
-	-

Example:

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

exti_flag_get

The description of exti_flag_get is shown as below:

Table 3-292. Function exti_flag_get

Function name	exti_flag_get
Function prototype	FlagStatus exti_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-281. exti_line_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

exti_flag_clear

The description of exti_flag_clear is shown as below:

Table 3-293. Function exti_flag_clear

Function name	exti_flag_clear
Function prototype	void exti_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-281. exti_line_enum
Output parameter{out}	

-	-
Return value	
-	-

Example:

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

```
exti_flag_clear(EXTI_0);
```

exti_interrupt_flag_get

The description of exti_interrupt_flag_get is shown as below:

Table 3-294. Function exti_interrupt_flag_get

Function name	exti_interrupt_flag_get
Function prototype	FlagStatus exti_interrupt_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-281. exti_line_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

```
FlagStatus state = exti_interrupt_flag_get(EXTI_0);
```

exti_interrupt_flag_clear

The description of exti_interrupt_flag_clear is shown as below:

Table 3-295. Function exti_interrupt_flag_clear

Function name	exti_interrupt_flag_clear
Function prototype	void exti_interrupt_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to Table 3-281. exti_line_enum
Output parameter{out}	
-	-

Return value	
-	-

Example:

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

```
exti_interrupt_flag_clear(EXTI_0);
```

3.11. FMC

There is flash controller and option byte for GD32A513 series. The FMC registers are listed in chapter [3.11.1](#) the FMC firmware functions are introduced in chapter [3.11.2](#).

3.11.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

Table 3-296. FMC Registers

Registers	Descriptions
FMC_WS	FMC wait state register
FMC_ECCCS	FMC ECC control and status register
FMC_KEY0	FMC unlock key register 0
FMC_STAT0	FMC status register 0
FMC_CTL0	FMC control register 0
FMC_ADDR0	FMC address register 0
FMC_OBKEY	FMC option byte unlock key register
FMC_KEY1	FMC unlock key register 1
FMC_STAT1	FMC status register 1
FMC_CTL1	FMC control register 1
FMC_ADDR1	FMC address register 1
FMC_EPCNT	FMC EEPROM counter register
FMC_OBSTAT	FMC option byte status register
FMC_WP0	FMC erase/program protection register 0
FMC_WP1	FMC erase/program protection register 1
FMC_OB1CS	FMC option byte 1 control and status register
FMC_PID	FMC product ID register

3.11.2. Descriptions of Peripheral functions

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

Table 3-297. FMC firmware function

Function name	Function description
fmc_unlock	unlock the main flash operation

Function name	Function description
fmc_bank0_unlock	unlock the main flash bank0 operation
fmc_bank1_unlock	unlock the main flash bank1 operation
fmc_lock	lock the main flash operation
fmc_bank0_lock	lock the main flash bank0 operation
fmc_bank1_lock	lock the main flash bank1 operation
fmc_wscnt_set	set the wait state counter value
fmc_prefetch_enable	enable pre-fetch
fmc_prefetch_disable	disable pre-fetch
fmc_cache_enable	enable cache
fmc_cache_disable	disable cache
fmc_cache_reset_enable	enable cache reset if cache is disabled
fmc_cache_reset_disable	disable cache reset
fmc_powerdown_mode_set	flash goto power-down mode when MCU enters deepsleep mode
fmc_sleep_mode_set	flash goto sleep mode when MCU enters deepsleep mode
fmc_sram_mode_config	configure shared SRAM mode
fmc_sram_mode_get	get shared SRAM mode
fmc_blank_check	check whether flash page is blank or not by check blank command
fmc_page_erase	erase main flash page
fmc_bank0_mass_erase	erase flash bank0
fmc_bank1_mass_erase	erase flash bank1
fmc_dflash_mass_erase	erase the data flash
fmc_mass_erase	erase whole chip
fmc_doubleword_program	program a double word at the corresponding address in main flash
fmc_fast_program	FMC fast program one row data (32 double-word) starting at the corresponding address
otp_doubleword_program	program a double word at the corresponding address in OTP
eeeprom_word_program	program a word at the corresponding address in EEPROM
eeeprom_word_read	read a word at the corresponding address in EEPROM
ob_unlock	unlock the option bytes 0 operation
ob_lock	lock the option bytes 0 operation
ob_reset	force to reload the option bytes 0
ob_erase	erase the option bytes 0
ob_write_protection_enable	enable option bytes 0 write protection
ob_security_protection_config	configure security protection
ob_user_write	program the FMC user option bytes
ob_data_program	program the FMC data option bytes
ob_user_get	get the value of FMC option bytes OB_USER in FMC_OBSTAT register

Function name	Function description
ob_data_get	get the value of FMC option bytes OB_DATA in FMC_OBSTAT register
ob_write_protection_get	get the value of FMC option bytes BK0WP in FMC_WP0 register
ob_bk1_write_protection_get	get the value of FMC option bytes BK1WP in FMC_WP1 register
ob_df_write_protection_get	get the value of FMC option bytes DFWP in FMC_WP1 register
ob_ep_write_protection_get	get the value of FMC option bytes EPWP in FMC_WP1 register
ob_plevel_get	get the value of FMC option bytes 0 security protection level (PLEVEL) in FMC_OBSTAT register
ob1_lock_config	configure lock value in option bytes 1
ob1_epload_config	configure the EPLOAD value of option bytes 1 loaded after the system reset
ob1_eeprom_parameter_config	configure option bytes 1 EEPROM parameters
dflash_size_get	get data flash size in byte unit
eeprom_backup_size_get	get EEPROM backup size in byte unit
eeprom_size_get	get EEPROM size in byte unit
fmc_flag_get	get FMC flag status
fmc_flag_clear	clear the FMC flag status
fmc_interrupt_enable	enable FMC interrupt
fmc_interrupt_disable	disable FMC interrupt
fmc_interrupt_flag_get	get FMC interrupt flag status
fmc_interrupt_flag_clear	clear FMC interrupt flag status

Enum fmc_state_enum

Table 3-298. fmc_state_enum

enum name	enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_PGSERR	program sequence error
FMC_PGERR	program error
FMC_PGAERR	program alignment error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error
FMC_CBCMDERR	the checked area not blank error
FMC_RSTERR	BOR/POR or system reset during flash erase/program error
FMC_OB_HSPC	FMC is under high security protection
FMC_OB1_LK	option bytes 1 is locked

Enum `fmc_sram_mode_enum`

Table 3-299. `fmc_sram_mode_enum`

enum name	enum description
<code>NO_SRAM_MODE</code>	SRAM mode is not configured
<code>FASTPG_SRAM_MODE</code>	fast program SRAM mode
<code>BASIC_SRAM_MODE</code>	basic SRAM mode
<code>EEPROM_SRAM_MODE</code>	EEPROM SRAM mode

Enum `fmc_area_enum`

Table 3-300. `fmc_area_enum`

enum name	enum description
<code>BANK0_AREA</code>	main flash bank0 area
<code>BANK1_AREA</code>	main flash bank1 area
<code>DATA_FLASH_AREA</code>	data flash area
<code>EEPROM_AREA</code>	EEPROM area

Enum `fmc_flag_enum`

Table 3-301. `fmc_flag_enum`

enum name	enum description
<code>FMC_BANK0_FLAG_BUSY</code>	flash bank0 busy flag
<code>FMC_BANK0_FLAG_PGSEERR</code>	flash bank0 program sequence error flag
<code>FMC_BANK0_FLAG_PGERR</code>	flash bank0 program error flag
<code>FMC_BANK0_FLAG_PGAERR</code>	flash bank0 program alignment error flag
<code>FMC_BANK0_FLAG_WPERR</code>	flash bank0 erase/program protection error flag
<code>FMC_BANK0_FLAG_END</code>	flash bank0 end of operation flag
<code>FMC_BANK0_FLAG_CBCMDERR</code>	flash bank0 checked area by the check blank command is all 0xFF or not flag
<code>FMC_BANK0_FLAG_RSTERR</code>	flash bank0 BOR/POR or system reset during erase/program flag
<code>FMC_BANK1_FLAG_BUSY</code>	flash bank1 busy flag

enum name	enum description
FMC_BANK1_FLAG_PGSEERR	flash bank1 program sequence error flag
FMC_BANK1_FLAG_PGERR	flash bank1 program error flag
FMC_BANK1_FLAG_PGAERR	flash bank1 program alignment error flag
FMC_BANK1_FLAG_WPERR	flash bank1 erase/program protection error flag
FMC_BANK1_FLAG_END	flash bank1 end of operation flag
FMC_BANK1_FLAG_CBCMDERR	flash bank1 checked area by the check blank command is all 0xFF or not flag
FMC_BANK1_FLAG_RSTERR	flash bank1 BOR/POR or system reset during erase/program flag
FMC_FLAG_OB0ECC	an ECC bit error is detected in option byte 0 flag
FMC_FLAG_BK1ECC	an ECC bit error is detected in bank 1 flag
FMC_FLAG_SYSECC	an ECC bit error is detected in system memory flag
FMC_FLAG_DFEC	an ECC bit error is detected in data flash flag
FMC_FLAG_OTPECC	an ECC bit error is detected in OTP flag
FMC_FLAG_OB1ECCDET	option bytes 1 two bit errors detect flag
FMC_FLAG_OB0ECCDET	option bytes 0 two bit errors detect flag
FMC_FLAG_EPECDET	EEPROM two bit errors detect flag
FMC_FLAG_ECCCOR	one bit error detected and correct flag
FMC_FLAG_ECCDET	OTP/data flash/system memory/bank1 two bit error detect flag
FMC_FLAG_OBERR	option bytes 0 error flag
FMC_FLAG_OB1ERR	option bytes 1 read error flag

Enum fmc_interrupt_flag_enum

Table 3-302. fmc_interrupt_flag_enum

enum name	enum description
FMC_BANK0_INT_FLAG_PGSERR	flash bank0 program sequence error interrupt flag
FMC_BANK0_INT_FLAG_PGERR	flash bank0 program error interrupt flag
FMC_BANK0_INT_FLAG_PGAERR	flash bank0 program alignment error interrupt flag
FMC_BANK0_INT_FLAG_WPERR	flash bank0 erase/program protection error interrupt flag
FMC_BANK0_INT_FLAG_END	flash bank0 end of operation interrupt flag
FMC_BANK0_INT_FLAG_CBCMDERR	flash bank0 checked area by the check blank command is all 0xFF or not interrupt flag
FMC_BANK0_INT_FLAG_RSTERR	flash bank0 BOR/POR or system reset during erase/program interrupt flag
FMC_BANK1_INT_FLAG_PGSERR	flash bank1 program sequence error interrupt flag
FMC_BANK1_INT_FLAG_PGERR	flash bank1 program error interrupt flag
FMC_BANK1_INT_FLAG_PGAERR	flash bank1 program alignment error interrupt flag
FMC_BANK1_INT_FLAG_WPERR	flash bank1 erase/program protection error interrupt flag
FMC_BANK1_INT_FLAG_END	flash bank1 end of operation interrupt flag
FMC_BANK1_INT_FLAG_CBCMDERR	flash bank1 checked area by the check blank command is all 0xFF or not interrupt flag
FMC_BANK1_INT_FLAG_RSTERR	flash bank1 BOR/POR or system reset during erase/program interrupt flag
FMC_INT_FLAG_OB1ECCDET	option bytes 1 two bit errors detect interrupt flag
FMC_INT_FLAG_OB0ECCDET	option bytes 0 two bit errors detect interrupt flag
FMC_INT_FLAG_EPECCDET	EEPROM two bit errors detect interrupt flag
FMC_INT_FLAG_ECCCOR	one bit error detected and correct interrupt flag
FMC_INT_FLAG_ECCDET	two bit errors detect interrupt flag

Enum fmc_interrupt_enum

Table 3-303. fmc_interrupt_enum

enum name	enum description
FMC_BANK0_INT_ERR	FMC bank0 error interrupt
FMC_BANK0_INT_END	FMC bank0 end of operation interrupt
FMC_BANK1_INT_ERR	FMC bank1 error interrupt
FMC_BANK1_INT_END	FMC bank1 end of operation interrupt
FMC_INT_ECCCOR	FMC one bit error correct interrupt
FMC_INT_ECCDET	FMC two bit errors interrupt

fmc_unlock

The description of fmc_unlock is shown as below:

Table 3-304. Function fmc_unlock

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

Example:

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

fmc_bank0_unlock

The description of fmc_bank0_unlock is shown as below:

Table 3-305. Function fmc_bank0_unlock

Function name	fmc_bank0_unlock
Function prototype	void fmc_bank0_unlock(void);

Function descriptions	unlock the main flash bank0 operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the main flash bank0 operation */
```

```
fmc_bank0_unlock();
```

fmc_bank1_unlock

The description of fmc_bank1_unlock is shown as below:

Table 3-306. Function fmc_bank1_unlock

Function name	fmc_bank1_unlock
Function prototype	void fmc_bank1_unlock(void);
Function descriptions	unlock the main flash bank1 operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* unlock the main flash bank1 operation */
```

```
fmc_bank1_unlock();
```

fmc_lock

The description of fmc_lock is shown as below:

Table 3-307. Function fmc_lock

Function name	fmc_lock
Function prototype	void fmc_lock(void);
Function descriptions	lock the main flash operation

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

Example:

```
/* lock the main flash operation */
```

```
fmc_lock();
```

fmc_bank0_lock

The description of fmc_bank0_lock is shown as below:

Table 3-308. Function fmc_bank0_lock

Function name	fmc_bank0_lock
Function prototype	void fmc_bank0_lock(void);
Function descriptions	lock the main flash bank0 operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the main flash bank0 operation */
```

```
fmc_bank0_lock();
```

fmc_bank1_lock

The description of fmc_bank1_lock is shown as below:

Table 3-309. Function fmc_bank1_lock

Function name	fmc_bank1_lock
Function prototype	void fmc_bank1_lock(void);
Function descriptions	lock the main flash bank1 operation
Precondition	-

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

Example:

```
/* lock the main flash bank1 operation */
```

```
fmc_bank1_lock();
```

fmc_wscnt_set

The description of fmc_wscnt_set is shown as below:

Table 3-310. Function fmc_wscnt_set

Function name	fmc_wscnt_set
Function prototype	void fmc_wscnt_set(uint32_t wscnt);
Function descriptions	set the wait state counter value
Precondition	-
The called functions	-
Input parameter{in}	
wscnt	wait state counter value
WS_WSCNT_0	0 wait state added
WS_WSCNT_1	1 wait state added
WS_WSCNT_2	2 wait state added
WS_WSCNT_3	3 wait state added
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the wait state counter value */
```

```
fmc_wscnt_set(WS_WSCNT_1);
```

fmc_prefetch_enable

The description of fmc_prefetch_enable is shown as below:

Table 3-311. Function fmc_prefetch_enable

Function name	fmc_prefetch_enable
---------------	---------------------

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

Example:

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

fmc_prefetch_disable

The description of fmc_prefetch_disable is shown as below:

Table 3-312. Function fmc_prefetch_disable

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

Example:

```
/* disable pre-fetch */
fmc_prefetch_disable();
```

fmc_cache_enable

The description of fmc_cache_enable is shown as below:

Table 3-313. Function fmc_cache_enable

Function name	fmc_cache_enable
Function prototype	void fmc_cache_enable(void);

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

Example:

```
/* enable cache */
```

```
fmc_cache_enable();
```

fmc_cache_disable

The description of fmc_cache_disable is shown as below:

Table 3-314. Function fmc_cache_disable

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

Example:

```
/* disable cache */
```

```
fmc_cache_disable();
```

fmc_cache_reset_enable

The description of fmc_cache_reset_enable is shown as below:

Table 3-315. Function fmc_cache_reset_enable

Function name	fmc_cache_reset_enable
Function prototype	void fmc_cache_reset_enable(void);
Function descriptions	enable cache reset if cache is disabled

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

Example:

```
/* enable cache reset if cache is disabled */
```

```
fmc_cache_reset_enable();
```

fmc_cache_reset_disable

The description of fmc_cache_reset_disable is shown as below:

Table 3-316. Function fmc_cache_reset_disable

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

Example:

```
/* disable cache reset */
```

```
fmc_cache_reset_disable();
```

fmc_powerdown_mode_set

The description of fmc_powerdown_mode_set is shown as below:

Table 3-317. Function fmc_powerdown_mode_set

Function name	fmc_powerdown_mode_set
Function prototype	void fmc_powerdown_mode_set(void);
Function descriptions	flash goto power-down mode when MCU enters deepsleep mode
Precondition	-

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

Example:

```
/* flash goto power-down mode when MCU enters deepsleep mode */
```

```
fmc_powerdown_mode_set();
```

fmc_sleep_mode_set

The description of fmc_sleep_mode_set is shown as below:

Table 3-318. Function fmc_sleep_mode_set

Function name	fmc_sleep_mode_set
Function prototype	void fmc_sleep_mode_set(void);
Function descriptions	flash goto sleep mode when MCU enters deepsleep mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* flash goto sleep mode when MCU enters deepsleep mode */
```

```
fmc_sleep_mode_set();
```

fmc_sram_mode_config

The description of fmc_sram_mode_config is shown as below:

Table 3-319. Function fmc_sram_mode_config

Function name	fmc_sram_mode_config
Function prototype	void fmc_sram_mode_config(fmc_sram_mode_enum sram_mode);
Function descriptions	configure shared SRAM mode
Precondition	-
The called functions	-

Input parameter{in}	
sram_mode	shared SRAM mode
<i>FASTPG_SRAM_MODE</i> <i>E</i>	fast program SRAM mode
<i>BASIC_SRAM_MODE</i>	basic SRAM mode
<i>EEPROM_SRAM_MODE</i> <i>E</i>	EEPROM SRAM mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure shared SRAM as fast PG SRAM */
fmc_sram_mode_config(FASTPG_SRAM_MODE);
```

fmc_sram_mode_get

The description of fmc_sram_mode_get is shown as below:

Table 3-320. Function fmc_sram_mode_get

Function name	fmc_sram_mode_get
Function prototype	fmc_sram_mode_enum fmc_sram_mode_get(void);
Function descriptions	get shared SRAM mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
fmc_sram_mode_enum	shared SRAM mode
<i>FASTPG_SRAM_MODE</i> <i>E</i>	fast program SRAM mode
<i>BASIC_SRAM_MODE</i>	basic SRAM mode
<i>EEPROM_SRAM_MODE</i> <i>E</i>	EEPROM SRAM mode
Return value	
-	-

Example:

```
/* get shared SRAM mode */
fmc_sram_mode_enum mode;
```

```
mode = fmc_sram_mode_get();
```

fmc_blank_check

The description of fmc_blank_check is shown as below:

Table 3-321. Function fmc_blank_check

Function name	fmc_blank_check
Function prototype	fmc_state_enum fmc_blank_check(uint32_t address, uint8_t length);
Function descriptions	check whether flash page is blank or not by check blank command
Precondition	-
The called functions	-
Input parameter{in}	
address	start address to check
Input parameter{in}	
length	the read length is 2^length double words, the flash area to be checked must be in one page and should not exceed 1KB boundary
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSEERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
/* check whether flash page is blank or not by check blank command */
```

```
fmc_state_enum state;
```

```
state = fmc_blank_check(0x8004000, 4);
```

fmc_page_erase

The description of fmc_page_erase is shown as below:

Table 3-322. Function fmc_page_erase

Function name	fmc_page_erase
Function prototype	fmc_state_enum fmc_page_erase(uint32_t page_address);

Function descriptions	erase main flash page
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
page_address	the page address to be erased
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
fmc_unlock();
```

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

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

fmc_bank0_mass_erase

The description of fmc_bank0_mass_erase is shown as below:

Table 3-323. Function fmc_bank0_mass_erase

Function name	fmc_bank0_mass_erase
Function prototype	fmc_state_enum fmc_bank0_mass_erase(void);
Function descriptions	erase flash bank0
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress

<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
fmc_unlock();
```

```
/* erase flash bank0 */
```

```
fmc_state_enum state = fmc_bank0_mass_erase();
```

fmc_bank1_mass_erase

The description of fmc_bank1_mass_erase is shown as below:

Table 3-324. Function fmc_bank1_mass_erase

Function name	fmc_bank1_mass_erase
Function prototype	fmc_state_enum fmc_bank1_mass_erase(void);
Function descriptions	erase flash bank1
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
fmc_unlock();
```

```
/* erase flash bank1 */
```



```
fmc_state_enum state = fmc_bank1_mass_erase();
```

fmc_dflash_mass_erase

The description of fmc_dflash_mass_erase is shown as below:

Table 3-325. Function fmc_dflash_mass_erase

Function name	fmc_dflash_mass_erase
Function prototype	fmc_state_enum fmc_dflash_mass_erase(void);
Function descriptions	erase the data flash
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
fmc_unlock();
```

```
/* erase the data flash */
```

```
fmc_state_enum state = fmc_dflash_mass_erase();
```

fmc_mass_erase

The description of fmc_mass_erase is shown as below:

Table 3-326. 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
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
fmc_unlock();

/* erase whole chip */

fmc_state_enum state = fmc_mass_erase();
```

fmc_doubleword_program

The description of fmc_doubleword_program is shown as below:

Table 3-327. Function fmc_doubleword_program

Function name	fmc_doubleword_program
Function prototype	fmc_state_enum fmc_doubleword_program(uint32_t address, uint64_t data);
Function descriptions	program a double word at the corresponding address in main flash
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
address	the address to be programmed
Input parameter{in}	
data	the data to be programmed
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress

<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
fmc_unlock();
```

```
fmc_page_erase(0x08004000);
```

```
/* program a double word at the corresponding address in main flash */
```

```
fmc_state_enum fmc_state = fmc_doubleword_program(0x08004000, 0x11223344aabbccdd);
```

fmc_fast_program

The description of fmc_fast_program is shown as below:

Table 3-328. Function fmc_fast_program

Function name	fmc_fast_program
Function prototype	fmc_state_enum fmc_fast_program(uint32_t address, uint64_t data[]);
Function descriptions	FMC fast program one row data (32 double-word) starting at the corresponding address
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
address	the address to be programmed
Input parameter{in}	
data	the data to be programmed
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error

FMC_RSTERR

BOR/POR or system reset during flash erase/program error

Example:

```

/* data buffer for fast programming */

static uint64_t data_buffer[32] = {

    0x0000000000000000U, 0x1111111111111111U, 0x2222222222222222U, 0x3333333333
33333333U,

    0x4444444444444444U, 0x5555555555555555U, 0x6666666666666666U, 0x77777777
77777777U,

    0x8888888888888888U, 0x9999999999999999U, 0xAAAAAAAAAAAAAAAAAU, 0xBB
BBBBBBBBBBBBBBU,

    0xCCCCCCCCCCCCCCCCU, 0xDDDDDDDDDDDDDDDDU, 0xEEEEEEEEEEEEEEEE
EEU, 0xFFFFFFFFFFFUFFFU,

    0x0011001100110011U, 0x2233223322332233U, 0x4455445544554455U, 0x667766
7766776677U,

    0x8899889988998899U, 0xAABBAABBAABBAABBU, 0xCCDDCCDDCCDDCCDDU,
0xEEFFEEFFEEFFEEFFU,

    0x2200220022002200U, 0x3311331133113311U, 0x6644664466446644U, 0x775577
5577557755U,

    0xAA88AA88AA88AA88U, 0xBB99BB99BB99BB99U, 0xEECCEECCEECCEECCECU,
0xFFDDFFDDFFDDFFDDU

};

fmc_unlock();

fmc_page_erase(0x08004000);

/* program flash */

fmc_state_enum fmc_state = fmc_fast_program(0x08004000, data_buffer);

```

otp_doubleword_program

The description of otp_doubleword_program is shown as below:

Table 3-329. Function otp_doubleword_program

Function name	otp_doubleword_program
Function prototype	fmc_state_enum otp_doubleword_program(uint32_t address, uint64_t data);
Function descriptions	program a double word at the corresponding address in OTP
Precondition	fmc_unlock

The called functions	-
Input parameter{in}	
address	the address to be programmed
Input parameter{in}	
data	the data to be programmed
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
fmc_unlock();
```

```
/* program a double word at the corresponding address in OTP */
```

```
fmc_state_enum fmc_state = otp_doubleword_program (0x1FFF7000, 0x11223344aabbccdd);
```

eeprom_word_program

The description of eeprom_word_program is shown as below:

Table 3-330. Function eeprom_word_program

Function name	eeprom_word_program
Function prototype	fmc_state_enum eeprom_word_program(uint32_t address, uint32_t data);
Function descriptions	program a word at the corresponding address in EEPROM
Precondition	-
The called functions	-
Input parameter{in}	
address	the address to be programmed
Input parameter{in}	
data	the data to be programmed
Output parameter{out}	
-	-
Return value	

fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

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

```
fmc_state_enum fmc_state = eeprom_word_program(0x08C00000, 0xaabbccdd);
```

eeprom_word_read

The description of eeprom_word_read is shown as below:

Table 3-331. Function eeprom_word_read

Function name	eeprom_word_read
Function prototype	uint32_t eeprom_word_read(uint32_t address);
Function descriptions	read a word at the corresponding address in EEPROM
Precondition	-
The called functions	-
Input parameter{in}	
address	address to read
Output parameter{out}	
-	-
Return value	
uint32_t	word data readout (0x0 – 0xFFFF FFFF)

Example:

```
/* read a word at the corresponding address in EEPROM */
```

```
uint32_t data = eeprom_word_read(0x08C00000);
```

ob_unlock

The description of ob_unlock is shown as below:

Table 3-332. Function ob_unlock

Function name	ob_unlock
Function prototype	void ob_unlock(void);

Function descriptions	unlock the option bytes 0 operation
Precondition	fmc_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
fmc_unlock();
```

```
/* unlock the option bytes 0 operation */
```

```
ob_unlock();
```

ob_lock

The description of ob_lock is shown as below:

Table 3-333. Function ob_lock

Function name	ob_lock
Function prototype	void ob_lock(void);
Function descriptions	lock the option bytes 0 operation
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the option bytes 0 operation */
```

```
ob_lock();
```

ob_reset

The description of ob_reset is shown as below:

Table 3-334. Function ob_reset

Function name	ob_reset
Function prototype	void ob_reset(void);

Function descriptions	force to reload the option bytes 0
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* force to reload the option bytes 0 */
```

```
ob_reset();
```

ob_erase

The description of ob_erase is shown as below:

Table 3-335. Function ob_erase

Function name	ob_erase
Function prototype	void ob_erase(void);
Function descriptions	erase the option bytes 0
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error
<i>FMC_OB_HSPC</i>	FMC is under high security protection

Example:

```
fmc_unlock();
```



```
ob_unlock();

/* erase the option bytes 0 */

fmc_state_enum fmc_state = ob_erase();
```

ob_write_protection_enable

The description of ob_write_protection_enable is shown as below:

Table 3-336. Function ob_write_protection_enable

Function name	ob_write_protection_enable
Function prototype	fmc_state_enum ob_write_protection_enable(fmc_area_enum wp_area, uint32_t ob_wp);
Function descriptions	enable option bytes 0 write protection
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
wp_area	write protection area, refer to Table 3-300. fmc_area_enum .
Input parameter{in}	
ob_wp	write protection configuration data. Notice that set the bit to 1 if you want to protect the corresponding pages. The lowest 8 bits is valid in area except bank0.
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSEERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error
<i>FMC_OB_HSPC</i>	FMC is under high security protection

Example:

```
fmc_unlock();

ob_unlock();

/* enable option bytes 0 write protection */

fmc_state_enum fmc_state = ob_write_protection_enable(BANK0_AREA, 0x00100000);
```

ob_security_protection_config

The description of ob_security_protection_config is shown as below:

Table 3-337. Function ob_security_protection_config

Function name	ob_security_protection_config
Function prototype	fmc_state_enum ob_security_protection_config(uint16_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_LSPC</i>	low security protection
<i>FMC_HSPC</i>	high security protection
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* enable security protection */

fmc_state = ob_security_protection_config(FMC_NSPC);
```

ob_user_write

The description of ob_user_write is shown as below:

Table 3-338. Function ob_user_write

Function name	ob_user_write
----------------------	---------------

Function prototype	fmc_state_enum ob_user_write(uint16_t ob_user);
Function descriptions	program the FMC user option bytes
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_user	user option bytes
OB_FWDGT_HW/OB_FWDGT_SW	free watchdog mode
OB_DEEPSLEEP_RST/OB_DEEPSLEEP_N_RST	generate a reset or enter deep-sleep mode
OB_STDBY_RST/OB_STDBY_N_RST	generate a reset or enter standby mode
OB_BOOT_FROM_BANK1/OB_BOOT_FROM_BANK0	boot mode
OB_BOOT_OTA_ENABLE/OB_BOOT_OTA_DISABLE	OTA mode
OB_BOR_DISABLE/OB_BOR_ENABLE	BOR on/off
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_PGSERR	program sequence error
FMC_PGERR	program error
FMC_PGAERR	program alignment error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error
FMC_CBCMDERR	the checked area not blank error
FMC_RSTERR	BOR/POR or system reset during flash erase/program error
FMC_OB_HSPC	FMC is under high security protection

Example:

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* program the FMC user option bytes */
```

```
fmc_state_enum fmc_state = ob_user_write(OB_FWDGT_HW | OB_BOOT_FROM_BANK0);
```

K0);

ob_data_program

The description of ob_data_program is shown as below:

Table 3-339. Function ob_data_program

Function name	ob_data_program
Function prototype	fmc_state_enum ob_data_program(uint16_t ob_data);
Function descriptions	program the FMC data option bytes
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_data	the data to be programmed, OB_DATA[0:15]
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error
<i>FMC_OB_HSPC</i>	FMC is under high security protection

Example:

```
fmc_unlock();

ob_unlock();

/* program option bytes data */

fmc_state_enum fmc_state = ob_data_program(0xdd22);
```

ob_user_get

The description of ob_user_get is shown as below:

Table 3-340. Function ob_user_get

Function name	ob_user_get
Function prototype	uint8_t ob_user_get(void);
Function descriptions	get the value of FMC option bytes OB_USER in FMC_OBSTAT register

Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the FMC user option bytes values(0x00 – 0xFF)

Example:

```
/* get the value of FMC option bytes OB_USER in FMC_OBSTAT register */
```

```
uint8_t user = ob_user_get();
```

ob_data_get

The description of ob_data_get is shown as below:

Table 3-341. Function ob_data_get

Function name	ob_data_get
Function prototype	uint16_t ob_data_get(void);
Function descriptions	get the value of FMC option bytes OB_DATA in FMC_OBSTAT register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	the FMC data option bytes values(0x0 – 0xFFFF)

Example:

```
/* get the value of FMC option bytes OB_DATA in FMC_OBSTAT register */
```

```
uint16_t data = ob_data_get();
```

ob_write_protection_get

The description of ob_write_protection_get is shown as below:

Table 3-342. Function ob_write_protection_get

Function name	ob_write_protection_get
Function prototype	uint32_t ob_write_protection_get(void);
Function descriptions	get the value of FMC option bytes BK0WP in FMC_WP0 register
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	the option bytes BK0WP value (0x0 – 0xFFFF FFFF)

Example:

```
/* get the value of FMC option bytes BK0WP in FMC_WP0 register */
```

```
uint32_t wp = ob_write_protection_get();
```

ob_bk1_write_protection_get

The description of ob_bk1_write_protection_get is shown as below:

Table 3-343. Function ob_bk1_write_protection_get

Function name	ob_bk1_write_protection_get
Function prototype	uint8_t ob_bk1_write_protection_get(void);
Function descriptions	get the value of FMC option bytes BK1WP in FMC_WP1 register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the option bytes BK1WP value (0x0 – 0xFF)

Example:

```
/* get the value of FMC option bytes BK1WP in FMC_WP1 register */
```

```
uint8_t wp = ob_bk1_write_protection_get();
```

ob_df_write_protection_get

The description of ob_df_write_protection_get is shown as below:

Table 3-344. Function ob_df_write_protection_get

Function name	ob_df_write_protection_get
Function prototype	uint8_t ob_df_write_protection_get(void);
Function descriptions	get the value of FMC option bytes DFWP in FMC_WP1 register
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the option bytes DFWP value (0x0 – 0xFF)

Example:

```
/* get the value of FMC option bytes DFWP in FMC_WP1 register */
```

```
uint8_t wp = ob_df_write_protection_get();
```

ob_ep_write_protection_get

The description of ob_ep_write_protection_get is shown as below:

Table 3-345. Function ob_ep_write_protection_get

Function name	ob_ep_write_protection_get
Function prototype	uint8_t ob_ep_write_protection_get(void);
Function descriptions	get the value of FMC option bytes EPWP in FMC_WP1 register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the option bytes EPWP value (0x0 – 0xFF)

Example:

```
/* get the value of FMC option bytes EPWP in FMC_WP1 register */
```

```
uint8_t ob_ep_write_protection_get();
```

ob_plevel_get

The description of ob_plevel_get is shown as below:

Table 3-346. Function ob_plevel_get

Function name	ob_plevel_get
Function prototype	uint8_t ob_plevel_get(void);
Function descriptions	get the value of FMC option bytes 0 security protection level (PLEVEL) in FMC_OBSTAT register
Precondition	-
The called functions	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	the value of PLEVEL
OB_OBSTAT_PLEVEL_NO	no security protection
OB_OBSTAT_PLEVEL_LOW	low security protection
OB_OBSTAT_PLEVEL_HIGH	high security protection

Example:

```
/* get the FMC option bytes security protection level */
```

```
uint8_t spc = ob_plevel_get();
```

ob1_lock_config

The description of ob1_lock_config is shown as below:

Table 3-347. Function ob1_lock_config

Function name	ob1_lock_config
Function prototype	fmc_state_enum ob1_lock_config(uint32_t lk_value);
Function descriptions	configure lock value in option bytes 1
Precondition	-
The called functions	-
Input parameter{in}	
lk_value	the LK value to be programmed
OB1CS_OB1_LK	when configured as OB1CS_OB1_LK, the option bytes 1 cannot be modified any more
OB1CS_OB1_NOT_LK	option bytes 1 is not locked
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_PGSERR	program sequence error
FMC_PGERR	program error
FMC_PGAERR	program alignment error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error

<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error
<i>FMC_OB1_LK</i>	option bytes 1 is locked

Example:

```
/* configure lock value in option bytes 1 */
```

```
fmc_state_enum fmc_state = ob1_lock_config(OB1CS_OB1_NOT_LK);
```

ob1_epload_config

The description of ob1_epload_config is shown as below:

Table 3-348. Function ob1_epload_config

Function name	ob1_epload_config
Function prototype	fmc_state_enum ob1_epload_config(uint32_t epload);
Function descriptions	configure the EPLOAD value of option bytes 1 loaded after the system reset
Precondition	-
The called functions	-
Input parameter{in}	
epload	EPLOAD value to be programmed
<i>OB1CS_EPLOAD_NOT_LOAD_EPDATA</i>	shared SRAM is not loaded with valid EEPROM data during FMC reset
<i>OB1CS_EPLOAD_LOAD_EPDATA</i>	shared SRAM is loaded with valid EEPROM data during FMC reset
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error
<i>FMC_OB1_LK</i>	option bytes 1 is locked

Example:

```
/* configure epload in option bytes 1 */
```

```
fmc_state_enum fmc_state = ob1_epload_config(OB1CS_EPLOAD_NOT_LOAD_EPDATA);
```

ob1_eeprom_parameter_config

The description of ob1_eeprom_parameter_config is shown as below:

Table 3-349. Function ob1_eeprom_parameter_config

Function name	ob1_eeprom_parameter_config
Function prototype	fmc_state_enum ob1_eeprom_parameter_config(uint32_t efalc, uint32_t epsize);
Function descriptions	configure option bytes 1 EEPROM parameters
Precondition	-
The called functions	-
Input parameter{in}	
efalc	EFALC value to be programmed
OB1CS_DF_64K_EF_0K	data flash size is 64KB, EEPROM backup size is 0KB
OB1CS_DF_48K_EF_16K	data flash size is 48KB, EEPROM backup size is 16KB
OB1CS_DF_32K_EF_32K	data flash size is 32KB, EEPROM backup size is 32KB
OB1CS_DF_16K_EF_48K	data flash size is 16KB, EEPROM backup size is 48KB
OB1CS_DF_0K_EF_64K	data flash size is 0KB, EEPROM backup size is 64KB
OB1CS_DF_EF_INVALID	data flash and EEPROM backup are invalid
OB1CS_DF_32K_EF_0K	data flash size is 32KB, EEPROM backup size is 0KB
OB1CS_DF_8K_EF_24K	data flash size is 8KB, EEPROM backup size is 24KB
OB1CS_DF_0K_EF_32K	data flash size is 0KB, EEPROM backup size is 32KB
OB1CS_DF_16K_EF_16K	data flash size is 16KB, EEPROM backup size is 16KB
OB1CS_DF_24K_EF_8K	data flash size is 24KB, EEPROM backup size is 8KB
OB1CS_DF_16K_EF_0K	data flash size is 16KB, EEPROM backup size is 0KB
OB1CS_DF_4K_EF_12K	data flash size is 4KB, EEPROM backup size is 12KB
OB1CS_DF_0K_EF_16K	data flash size is 0KB, EEPROM backup size is 16KB
OB1CS_DF_8K_EF_8K	data flash size is 8KB, EEPROM backup size is 8KB
OB1CS_DF_12K_EF_4K	data flash size is 12KB, EEPROM backup size is 4KB

<i>K</i>	
Input parameter{in}	
epsize	EPSIZE value to be programmed
<i>OB1CS_EPSIZE_NON E</i>	no EEPROM
<i>OB1CS_EPSIZE_4K</i>	4KB EEPROM
<i>OB1CS_EPSIZE_2K</i>	2KB EEPROM
<i>OB1CS_EPSIZE_1K</i>	1KB EEPROM
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error
<i>FMC_RSTERR</i>	BOR/POR or system reset during flash erase/program error
<i>FMC_OB1_LK</i>	option bytes 1 is locked

Example:

```
/* configure option bytes 1 EEPROM parameters */
```

```
fmc_state_enum fmc_state = ob1_eeprom_parameter_config(OB1CS_DF_48K_EF_16K,
OB1CS_EPSIZE_4K);
```

dflash_size_get

The description of dflash_size_get is shown as below:

Table 3-350. Function dflash_size_get

Function name	dflash_size_get
Function prototype	uint32_t dflash_size_get(void);
Function descriptions	get data flash size in byte unit
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

uint32_t	data flash byte count (0x0 – 0xFFFF FFFF)
-----------------	---

Example:

```
/* get data flash size in byte unit */
```

```
uint32_t size = dflash_size_get();
```

eeeprom_backup_size_get

The description of eeeprom_backup_size_get is shown as below:

Table 3-351. Function eeeprom_backup_size_get

Function name	eeeprom_backup_size_get
Function prototype	uint32_t eeeprom_backup_size_get(void);
Function descriptions	get EEPROM backup size in byte unit
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	EEPROM backup byte count (0x0 – 0xFFFF FFFF)

Example:

```
/* get EEPROM backup size in byte unit */
```

```
uint32_t size = eeeprom_backup_size_get();
```

eeeprom_size_get

The description of eeeprom_size_get is shown as below:

Table 3-352. Function eeeprom_size_get

Function name	eeeprom_size_get
Function prototype	uint32_t eeeprom_size_get(void);
Function descriptions	get EEPROM size in byte unit
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	EEPROM byte count (0x0 – 0xFFFF FFFF)

Example:

```
/* get EEPROM size in byte unit */

uint32_t size = eeprom_size_get();
```

fmc_flag_get

The description of fmc_flag_get is shown as below:

Table 3-353. Function fmc_flag_get

Function name	fmc_flag_get
Function prototype	FlagStatus fmc_flag_get(fmc_flag_enum flag);
Function descriptions	get FMC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	FMC flag, refer to Table 3-301. fmc_flag_enum .
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get FMC end flag */

FlagStatus flag = fmc_flag_get(FMC_BANK0_FLAG_END);
```

fmc_flag_clear

The description of fmc_flag_clear is shown as below:

Table 3-354. Function fmc_flag_clear

Function name	fmc_flag_clear
Function prototype	void fmc_flag_clear(fmc_flag_enum flag);
Function descriptions	clear the FMC flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	FMC flag
<i>FMC_BANK0_FLAG_PGERR</i>	flash bank0 program sequence error flag
<i>FMC_BANK0_FLAG_PGERR</i>	flash bank0 program error flag
<i>FMC_BANK0_FLAG_PGERR</i>	FMC_BANK0_FLAG_PGERR

<i>FMC_BANK0_FLAG_W PERR</i>	flash bank0 erase/program protection error flag
<i>FMC_BANK0_FLAG_E ND</i>	flash bank0 end of operation flag
<i>FMC_BANK0_FLAG_C BCMDERR</i>	flash bank0 checked area by the check blank command is all 0xFF or not flag
<i>FMC_BANK0_FLAG_R STERR</i>	flash bank0 BOR/POR or system reset during erase/program flag
<i>FMC_BANK1_FLAG_P GSERR</i>	flash bank1 program sequence error flag
<i>FMC_BANK1_FLAG_P GERR</i>	flash bank1 program error flag
<i>FMC_BANK1_FLAG_P GAERR</i>	flash bank1 program alignment error flag
<i>FMC_BANK1_FLAG_W PERR</i>	flash bank1 erase/program protection error flag
<i>FMC_BANK1_FLAG_E ND</i>	flash bank1 end of operation flag
<i>FMC_BANK1_FLAG_C BCMDERR</i>	flash bank1 checked area by the check blank command is all 0xFF or not flag
<i>FMC_BANK1_FLAG_R STERR</i>	flash bank1 BOR/POR or system reset during erase/program flag
<i>FMC_FLAG_OB1ECC DET</i>	option bytes 1 two bit error detect flag
<i>FMC_FLAG_OB0ECC DET</i>	option bytes 0 two bit error detect flag
<i>FMC_FLAG_EPECCD ET</i>	EEPROM two bit errors detect flag
<i>FMC_FLAG_ECCCOR</i>	one bit error detected and correct flag
<i>FMC_FLAG_ECCDET</i>	OTP/data flash/system memory/bank1 two bit error detect flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear FMC end flag */
```

```
fmc_flag_clear(FMC_BANK0_FLAG_END);
```

fmc_interrupt_enable

The description of fmc_interrupt_enable is shown as below:

Table 3-355. Function `fmc_interrupt_enable`

Function name	<code>fmc_interrupt_enable</code>
Function prototype	<code>void fmc_interrupt_enable(fmc_interrupt_enum interrupt);</code>
Function descriptions	enable FMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	FMC interrupt, refer to Table 3-303. fmc_interrupt_enum .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable FMC end interrupt */
fmc_interrupt_enable(FMC_BANK0_INT_END);
```

`fmc_interrupt_disable`

The description of `fmc_interrupt_disable` is shown as below:

Table 3-356. Function `fmc_interrupt_disable`

Function name	<code>fmc_interrupt_disable</code>
Function prototype	<code>void fmc_interrupt_disable(fmc_interrupt_enum interrupt);</code>
Function descriptions	disable FMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	FMC interrupt, refer to Table 3-303. fmc_interrupt_enum .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable FMC end interrupt */
fmc_interrupt_disable(FMC_BANK0_INT_END);
```

`fmc_interrupt_flag_get`

The description of `fmc_interrupt_flag_get` is shown as below:

Table 3-357. Function `fmc_interrupt_flag_get`

Function name	<code>fmc_interrupt_flag_get</code>
Function prototype	<code>FlagStatus fmc_interrupt_flag_get(fmc_interrupt_flag_enum int_flag);</code>
Function descriptions	get FMC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
<code>int_flag</code>	FMC interrupt flag, refer to Table 3-302. fmc_interrupt_flag_enum .
Output parameter{out}	
-	-
Return value	
<code>FlagStatus</code>	SET or RESET

Example:

```
/* check FMC program operation error flag is set or not */
```

```
FlagStatus flag = fmc_interrupt_flag_get(FMC_BANK0_INT_FLAG_PGERR);
```

`fmc_interrupt_flag_clear`

The description of `fmc_interrupt_flag_get` is shown as below:

Table 3-358. Function `fmc_interrupt_flag_clear`

Function name	<code>fmc_interrupt_flag_clear</code>
Function prototype	<code>void fmc_interrupt_flag_clear(fmc_interrupt_flag_enum int_flag);</code>
Function descriptions	clear FMC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
<code>int_flag</code>	FMC interrupt flag, refer to Table 3-302. fmc_interrupt_flag_enum .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear FMC program operation error flag */
```

```
fmc_interrupt_flag_get(FMC_BANK0_INT_FLAG_PGERR);
```

3.12. FWDGT

The free watchdog timer (FWDGT) is a hardware timing circuitry that can be used to detect system failures due to software malfunctions. It's suitable for the situation that requires an

independent environment and lower timing accuracy. The FWDGT registers are listed in chapter [3.12.1](#) the FWDGT firmware functions are introduced in chapter [3.12.2](#).

3.12.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

Table 3-359. FWDGT Registers

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

3.12.2. Descriptions of Peripheral functions

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

Table 3-360. FWDGT firmware function

Function name	Function description
fwdgt_write_enable	enable write access to FWDGT_PSC and FWDGT_RLD and FWDGT_WND
fwdgt_write_disable	disable write access to FWDGT_PSC and FWDGT_RLD and FWDGT_WND
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_window_value_config	configure the FWDGT counter window 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 3-361. 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 and FWDGT_WND
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* enable write access to FWDGT_PSC and FWDGT_RLD and FWDGT_WND */
```

```
fwdgt_write_enable();
```

fwdgt_write_disable

The description of fwdgt_write_disable is shown as below:

Table 3-362. Function fwdgt_write_disable

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

Example:

```
/* disable write access to FWDGT_PSC,FWDGT_RLD and FWDGT_WND */
```

```
fwdgt_write_disable();
```

fwdgt_enable

The description of fwdgt_enable is shown as below:

Table 3-363. 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 */
```

```
fwdgt_enable();
```

fwdgt_prescaler_value_config

The description of fwdgt_prescaler_value_config is shown as below:

Table 3-364. Function fwdgt_prescaler_value_config

Function name	fwdgt_prescaler_value_config
Function prototype	ErrStatus fwdgt_prescaler_value_config(uint16_t prescaler_value);
Function descriptions	configure the FWDGT counter clock prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
prescaler_value	specify prescaler value
FWDGT_PSC_DIVx	FWDGT prescaler set to x(x=4,8,16,32,64,128,256)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```
/* set FWDGT prescaler to 4 */
```

```
ErrStatus flag;
```

```
flag = fwdgt_prescaler_value_config(FWDGT_PSC_DIV4);
```

fwdgt_reload_value_config

The description of fwdgt_reload_value_config is shown as below:

Table 3-365. Function fwdgt_reload_value_config

Function name	fwdgt_reload_value_config
Function prototype	ErrStatus fwdgt_reload_value_config(uint16_t reload_value);
Function descriptions	configure the FWDGT counter reload value
Precondition	-
The called functions	-
Input parameter{in}	

reload_value	reload_value, specify reload value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

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

ErrStatus flag;

```
flag = fwdgt_reload_value_config(0xFFF);
```

fwdgt_window_value_config

The description of fwdgt_window_value_config is shown as below:

Table 3-366. Function fwdgt_window_value_config

Function name	fwdgt_window_value_config
Function prototype	ErrStatus fwdgt_window_value_config(uint16_t window_value);
Function descriptions	configure the FWDGT counter window value
Precondition	-
The called functions	-
Input parameter{in}	
window_value	window_value, specify window value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```
/* set FWDGT window value to 0xFFF */
```

ErrStatus flag;

```
flag = fwdgt_window_value_config(0xFFF);
```

fwdgt_counter_reload

The description of fwdgt_counter_reload is shown as below:

Table 3-367. 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 3-368. Function fwdgt_config

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

Example:

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

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

fwdgt_flag_get

The description of fwdgt_flag_get is shown as below:

Table 3-369. 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
<i>FWDGT_FLAG_WUD</i>	a write operation to FWDGT_WND 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);
```

3.13. GPIO

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

3.13.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

Table 3-370. 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

Registers	Descriptions
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

3.13.2. Descriptions of Peripheral functions

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

Table 3-371. 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
gpio_pin_lock	lock GPIO pin bit
gpio_bit_toggle	toggle GPIO pin status
gpio_port_toggle	toggle GPIO port status

gpio_deinit

The description of gpio_deinit is shown as below:

Table 3-372. Function gpio_deinit

Function name	gpio_deinit
Function prototype	void gpio_deinit(uint32_t gpio_periph);
Function descriptions	reset GPIO port
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F)
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 3-373. 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
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F)
Input parameter{in}	
mode	gpio pin mode
<i>GPIO_MODE_INPUT</i>	input mode
<i>GPIO_MODE_OUTPUT</i>	output mode
<i>GPIO_MODE_AF</i>	alternate function mode
<i>GPIO_MODE_ANALOG</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>	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:

```
/* config 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 3-374. 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,D,E,F)
Input parameter{in}	
otype	gpio pin output mode
<i>GPIO_OTYPE_PP</i>	push pull mode
<i>GPIO_OTYPE_OD</i>	open drain mode
Input parameter{in}	
speed	gpio pin output max speed
<i>GPIO_OSPEED_2MHZ</i>	output max speed 2MHz
<i>GPIO_OSPEED_10MHZ</i>	output max speed 10MHz
<i>GPIO_OSPEED_50MHZ</i>	output max speed 50MHz
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:

```
/* config 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 3-375. 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
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F)
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 */
```

```
gpio_bit_set(GPIOA, GPIO_PIN_0);
```

gpio_bit_reset

The description of gpio_bit_reset is shown as below:

Table 3-376. Function gpio_bit_reset

Function name	gpio_bit_reset
Function prototype	void gpio_bit_reset(uint32_t gpio_periph,uint32_t pin);
Function descriptions	reset GPIO pin
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F)
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_set(GPIOA, GPIO_PIN_0);
```

gpio_bit_write

The description of gpio_bit_write is shown as below:

Table 3-377. 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,D,E,F)
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 3-378. Function gpio_port_write

Function name	gpio_port_write
Function prototype	void gpio_port_write(uint32_t gpio_periph,uint16_t data);
Function descriptions	write data to the specified GPIO port
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F)
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 3-379. Function gpio_input_bit_get

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

Example:

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

```
FlagStatus bit_state;
```

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

gpio_input_port_get

The description of gpio_input_port_get is shown as below:

Table 3-380. 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,D,E,F)
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 3-381. 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,D,E,F)
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 3-382. 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,D,E,F)
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 3-383. 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
GPIOx	GPIOx(x =A,B,C,D,E,F)
Input parameter{in}	
alt_func_num	GPIO pin af function, please refer to specific device datasheet
GPIO_AF_0	SYSTEM
GPIO_AF_1	TIMER0, TIMER1, TIMER7, TIMER19, TIMER20
GPIO_AF_2	TIMER0, TIMER1, TIMER7, TIMER19, TIMER20
GPIO_AF_3	TIMER7, TIMER19, I2C0
GPIO_AF_4	SPI0, SPI1, I2S1, USART1
GPIO_AF_5	USART0, USART2, MFCOM, SPI1, I2C1
GPIO_AF_6	CAN0, CAN1, MFCOM, TRIGSEL
GPIO_AF_7	TRIGSEL, CMP, MFCOM
GPIO_AF_8	-
GPIO_AF_9	EVENTOUT
Input parameter{in}	
pin	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	All pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*set PA0 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 3-384. 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
GPIOx	GPIOx(x = A,B,C,D,E,F)
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 3-385. 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,D,E,F)
Input parameter{in}	
pin	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	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 3-386. 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
GPIOx	GPIOx(x = A,B,C,D,E,F)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* toggle GPIOA */
gpio_port_toggle(GPIOA);
```

3.14. I2C

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. The I2C registers are listed in chapter [3.14.1](#), the I2C firmware functions are introduced in chapter [3.14.2](#).

3.14.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

Table 3-387. 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

Registers	Descriptions
I2C_CTL2	Control register 2

3.14.2. Descriptions of Peripheral functions

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

Table 3-388. 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
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_receved_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

Function name	Function description
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
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 3-389. 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

Member name	Function description
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	overrun/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 3-390. 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 3-391. 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
I2Cx	(x=0,1)
Input parameter{in}	
psc	0-0x0000000F, timing prescaler
Input parameter{in}	
scl_dely	0-0x0000000F,data setup time
Input parameter{in}	
sda_dely	0-0x0000000F,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 3-392. 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
I2Cx	(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 t _{I2CCLK}
<i>FILTER_LENGTH_2</i>	digital filter is enabled and filter spikes with a length of up to 2 t _{I2CCLK}
<i>FILTER_LENGTH_3</i>	digital filter is enabled and filter spikes with a length of up to 3 t _{I2CCLK}
<i>FILTER_LENGTH_4</i>	digital filter is enabled and filter spikes with a length of up to 4 t _{I2CCLK}
<i>FILTER_LENGTH_5</i>	digital filter is enabled and filter spikes with a length of up to 5 t _{I2CCLK}
<i>FILTER_LENGTH_6</i>	digital filter is enabled and filter spikes with a length of up to 6 t _{I2CCLK}
<i>FILTER_LENGTH_7</i>	digital filter is enabled and filter spikes with a length of up to 7 t _{I2CCLK}

<i>FILTER_LENGTH_8</i>	digital filter is enabled and filter spikes with a length of up to 8 t_{I2CCLK}
<i>FILTER_LENGTH_9</i>	digital filter is enabled and filter spikes with a length of up to 9 t_{I2CCLK}
<i>FILTER_LENGTH_10</i>	digital filter is enabled and filter spikes with a length of up to 10 t_{I2CCLK}
<i>FILTER_LENGTH_11</i>	digital filter is enabled and filter spikes with a length of up to 11 t_{I2CCLK}
<i>FILTER_LENGTH_12</i>	digital filter is enabled and filter spikes with a length of up to 12 t_{I2CCLK}
<i>FILTER_LENGTH_13</i>	digital filter is enabled and filter spikes with a length of up to 13 t_{I2CCLK}
<i>FILTER_LENGTH_14</i>	digital filter is enabled and filter spikes with a length of up to 14 t_{I2CCLK}
<i>FILTER_LENGTH_15</i>	digital filter is enabled and filter spikes with a length of up to 15 t_{I2CCLK}
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* I2C0 digital filter filters spikes with a length of up to 1  $t_{I2CCLK}$  */
```

```
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 3-393. 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 3-394. 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 3-395. 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 3-396. 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 3-397. 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 3-398. 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 3-399. 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 3-400. 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 3-401. 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
I2Cx	(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 3-402. 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
I2Cx	(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 3-403. 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 3-404. 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 3-405. 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 3-406. 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 3-407. 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 3-408. 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_COM</i>	address bit1 needs compare

<i>PARE</i>	
<i>ADDRESS_BIT2_COM</i> <i>PARE</i>	address bit2 needs compare
<i>ADDRESS_BIT3_COM</i> <i>PARE</i>	address bit3 needs compare
<i>ADDRESS_BIT4_COM</i> <i>PARE</i>	address bit4 needs compare
<i>ADDRESS_BIT5_COM</i> <i>PARE</i>	address bit5 needs compare
<i>ADDRESS_BIT6_COM</i> <i>PARE</i>	address bit6 needs compare
<i>ADDRESS_BIT7_COM</i> <i>PARE</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 3-409. 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 3-410. 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
ADDRESS2_NO_MAS K	no mask, all the bits must be compared
ADDRESS2_MASK_BIT1	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 3-411. 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
<i>I2Cx</i>	(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 3-412. 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	0x00000000..0x0000007F

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 3-413. 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 3-414. 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 3-415. 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 3-416. 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_enable

The description of i2c_enable is shown as below:

Table 3-417. 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 3-418. 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 3-419. 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 3-420. 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 3-421. 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 3-422. 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	0x00000000..0x000000FF

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 3-423. 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 3-424. 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 3-425. 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 3-426. 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 3-427. 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 3-428. 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
I2Cx	(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 3-429. 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
I2Cx	(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 3-430. 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
I2Cx	(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 3-431. 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
I2Cx	(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 3-432. 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 3-433. 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
<i>I2Cx</i>	(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 3-434. 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
<i>I2Cx</i>	(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 3-435. 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
<i>I2Cx</i>	(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 3-436. 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
<i>I2Cx</i>	(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 3-437. 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
<i>I2Cx</i>	(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 3-438. 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
<i>I2Cx</i>	(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 3-439. 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
<i>I2Cx</i>	(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 3-440. 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
<i>I2Cx</i>	(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 3-441. 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
<i>I2Cx</i>	(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 3-442. 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
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
timeout	0x00000000-0x00000FFF, 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 3-443. 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	0x00000000-0x00000FFF, 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 3-444. 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 3-445. 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
<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
flag	I2C flags
<i>I2C_FLAG_TBE</i>	I2C_TDATA is empty during transmitting
<i>I2C_FLAG_TI</i>	transmit interrupt
<i>I2C_FLAG_RBNE</i>	I2C_RDATA is not empty during receiving
<i>I2C_FLAG_ADDSEND</i>	address received matches in slave mode
<i>I2C_FLAG_NACK</i>	not acknowledge flag
<i>I2C_FLAG_STPDET</i>	STOP condition detected in slave mode
<i>I2C_FLAG_TC</i>	transfer complete in master mode
<i>I2C_FLAG_TCR</i>	transfer complete reload
<i>I2C_FLAG_BERR</i>	bus error
<i>I2C_FLAG_LOSTARB</i>	arbitration Lost
<i>I2C_FLAG_OUERR</i>	overrun/underrun error in slave mode
<i>I2C_FLAG_PECERR</i>	PEC error
<i>I2C_FLAG_TIMEOUT</i>	timeout flag
<i>I2C_FLAG_SMBALT</i>	SMBus Alert
<i>I2C_FLAG_I2CBSY</i>	busy flag
<i>I2C_FLAG_TR</i>	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 3-446. 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

<i>I2Cx</i>	(x=0,1)
Input parameter{in}	
flag	I2C flags
<i>I2C_FLAG_ADDSEND</i>	address received matches in slave mode
<i>I2C_FLAG_NACK</i>	not acknowledge flag
<i>I2C_FLAG_STPDET</i>	STOP condition detected in slave mode
<i>I2C_FLAG_BERR</i>	bus error
<i>I2C_FLAG_LOSTARB</i>	arbitration Lost
<i>I2C_FLAG_OUERR</i>	overflow/underrun error in slave mode
<i>I2C_FLAG_PECERR</i>	PEC error
<i>I2C_FLAG_TIMEOUT</i>	timeout flag
<i>I2C_FLAG_SMBALT</i>	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 3-447. Function i2c_interrupt_enable

Function name	i2c_interrupt_enable
Function prototype	void i2c_interrupt_enable(uint32_t i2c_periph, uint32_t interrupt);
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 3-448. Function i2c_interrupt_disable

Function name	i2c_interrupt_disable
Function prototype	void i2c_interrupt_disable(uint32_t i2c_periph, uint32_t interrupt);
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 3-449. 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
I2Cx	(x=0,1)
Input parameter{in}	
int_flag	I2C interrupt flags, refer to Table 3-389. i2c_interrupt_flag_enum .
I2C_INT_FLAG_TI	transmit interrupt flag
I2C_INT_FLAG_RBNE	I2C_RDATA is not empty during receiving interrupt flag
I2C_INT_FLAG_ADDS END	address received matches in slave mode interrupt flag
I2C_INT_FLAG_NACK	not acknowledge interrupt flag
I2C_INT_FLAG_STPD ET	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_LOSTA RB	arbitration lost interrupt flag
I2C_INT_FLAG_OUER R	overflow/underrun error in slave mode interrupt flag
I2C_INT_FLAG_PEC RR	PEC error interrupt flag
I2C_INT_FLAG_TIMEO UT	timeout interrupt flag
I2C_INT_FLAG_SMBA LT	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 3-450. 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 3-389. i2c_interrupt_flag_enum .
<i>I2C_INT_FLAG_ADDS 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 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 RB</i>	arbitration lost interrupt flag
<i>I2C_INT_FLAG_OUER R</i>	overflow/underrun error in slave mode interrupt flag
<i>I2C_INT_FLAG_PECE RR</i>	PEC error interrupt flag
<i>I2C_INT_FLAG_TIMEO UT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBA LT</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);
```

3.15. MFCOM

The MFCOM is a highly configurable module provide emulation of a variety of serial communication protocols and flexible timers. The PMU registers are listed in chapter [3.15.1](#), the PMU firmware functions are introduced in chapter [3.15.2](#).

3.15.1. Descriptions of Peripheral registers

MFCOM registers are listed in the table shown as below:

Table 3-451. MFCOM Registers

Registers	Descriptions
MFCOM_CTL	Control register
MFCOM_PINDATA	Pin data register
MFCOM_SSTAT	Shifter status register
MFCOM_SERR	Shifter error register
MFCOM_TMSTAT	Timer status register
MFCOM_SSIEN	Shifter status interrupt enable register
MFCOM_SEIEN	Shifter error interrupt enable register
MFCOM_TMSIEN	Timer status interrupt enable register
MFCOM_SSDMAEN	Shifter status DMA enable register
MFCOM_SCTLx	Shifter control x register
MFCOM_SCFGx	Shifter configuration x register
MFCOM_SBUFx	Shifter buffer x register
MFCOM_SBUFBISx	Shifter buffer x bit swapped register
MFCOM_SBUFBYSx	Shifter buffer x byte swapped register
MFCOM_SBUFBBSx	Shifter buffer x bit byte swapped register
MFCOM_TMCTLx	Timer control x register
MFCOM_TMCFGx	Timer configuration x register
MFCOM_TMCMPx	Timer compare x register

3.15.2. Descriptions of Peripheral functions

MFCOM ware functions are listed in the table shown as below:

Table 3-452. MFCOM firmware function

Function name	Function description
mfcom_deinit	reset most part of MFCOM register
mfcom_software_reset	software reset
mfcom_enable	enable MFCOM function
mfcom_disable	disable MFCOM function
mfcom_timer_struct_para_init	initialize mfcom_timer_parameter_struct with the default values

Function name	Function description
mfcom_shifter_struct_para_init	initialize mfcom_shifter_parameter_struct with the default values
mfcom_timer_init	initialize MFCOM timer
mfcom_shifter_init	initialize MFCOM shifter
mfcom_timer_pin_config	configure timer pin mode
mfcom_shifter_pin_config	configure shifter pin mode
mfcom_timer_enable	enable MFCOM timer in specific mode
mfcom_shifter_enable	enable MFCOM shifter in specific mode
mfcom_timer_disable	disable MFCOM timer
mfcom_shifter_disable	disable MFCOM shifter
mfcom_timer_cmpvalue_set	set the timer compare value
mfcom_timer_cmpvalue_get	get the timer compare value
mfcom_timer_dismode_set	set the timer disable source
mfcom_shifter_stopbit_set	set the shifter stopbit
mfcom_buffer_write	write MFCOM shifter buffer
mfcom_buffer_read	read MFCOM shifter buffer
mfcom_shifter_flag_get	get MFCOM shifter flag
mfcom_shifter_error_flag_get	get MFCOM shifter error flag
mfcom_timer_flag_get	get MFCOM timer flag
mfcom_shifter_interrupt_flag_get	get MFCOM shifter interrupt flag
mfcom_shifter_error_interrupt_flag_get	get MFCOM shifter error interrupt flag
mfcom_timer_interrupt_flag_get	get MFCOM timer interrupt flag
mfcom_shifter_flag_clear	clear MFCOM shifter flag
mfcom_shifter_error_flag_clear	clear MFCOM shifter error flag
mfcom_timer_flag_clear	clear MFCOM timer flag
mfcom_shifter_interrupt_enable	enable MFCOM shifter interrupt
mfcom_shifter_error_interrupt_enable	enable MFCOM shifter error interrupt
mfcom_timer_interrupt_enable	enable MFCOM timer interrupt
mfcom_shifter_dma_enable	enable MFCOM shifter dma
mfcom_shifter_interrupt_disable	disable MFCOM shifter interrupt
mfcom_shifter_error_interrupt_disable	disable MFCOM shifter error interrupt
mfcom_timer_interrupt_disable	disable MFCOM timer interrupt
mfcom_shifter_dma_disable	disable MFCOM shifter dma

Struct mfcom_timer_parameter_struct

Table 3-453. Struct mfcom_timer_parameter_struct

Member name	Function description
trigger_select	the internal trigger selection
trigger_polarity	trigger polarity
pin_config	timer pin configuration
pin_select	timer pin number select

pin_polarity	timer pin polarity
mode	timer work mode
output	configures the initial state of the timer output and whether it is affected by the timer reset
decrement	configures the source of the timer decrement and the source of the shift clock
reset	configures the condition that causes the timer counter (and optionally the timer output) to be reset
disable	configures the condition that causes the timer to be disabled and stop decrementing
enable	configures the condition that causes the timer to be enabled and start decrementing
stopbit	timer stop bit generation
startbit	timer start bit generation
compare	value for timer compare x register

Struct mfcom_shifter_parameter_struct

Table 3-454. Struct mfcom_shifter_parameter_struct

Member name	Function description
timer_select	selects which timer is used for controlling the logic/shift register and generating the shift clock
timer_polarity	timer polarity
pin_config	shifter pin configuration
pin_select	shifter pin number select
pin_polarity	shifter pin polarity
mode	configures the mode of the shifter
input_source	selects the input source for the shifter
stopbit	shifter stop bit
startbit	shifter start bit

mfcom_deinit

The description of mfcom_deinit is shown as below:

Table 3-455. Function mfcom_deinit

Function name	mfcom_deinit
Function prototype	void mfcom_deinit(void);
Function descriptions	reset MFCOM
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* reset most of the MFCOM registers*/
```

```
mfcom_deinit();
```

mfcom_software_reset

The description of mfcom_software_reset is shown as below:

Table 3-456. Function mfcom_software_reset

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

Example:

```
/* software reset */
```

```
mfcom_software_reset();
```

mfcom_enable

The description of mfcom_enable is shown as below:

Table 3-457. Function mfcom_enable

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

-	-
---	---

Example:

```
/* enable MFCOM function*/
```

```
mfcom_enable();
```

mfcom_disable

The description of mfcom_disable is shown as below:

Table 3-458. Function mfcom_disable

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

Example:

```
/* disable MFCOM function*/
```

```
mfcom_disable();
```

mfcom_timer_struct_para_init

The description of mfcom_timer_struct_para_init is shown as below:

Table 3-459. Function mfcom_timer_struct_para_init

Function name	mfcom_timer_struct_para_init
Function prototype	void mfcom_timer_struct_para_init(mfcom_timer_parameter_struct* init_struct);
Function descriptions	initialize mfcom_timer_parameter_struct with the default values
Precondition	-
The called function	-
Input parameter{in}	
init_struct	Mfcom timer parameter struct, the structure members can refer to members of the structure Struct mfcom timer parameter struct
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* initialize mfcom_timer_parameter_struct with the default values*/
```

```
mfcom_timer_parameter_struct mfcom_timer_struct
```

```
mfcom_timer_struct_para_init(&mfcom_timer_struct);
```

mfcom_shifter_struct_para_init

The description of mfcom_shifter_struct_para_init is shown as below:

Table 3-460. Function mfcom_shifter_struct_para_init

Function name	mfcom_shifter_struct_para_init
Function prototype	void mfcom_shifter_struct_para_init(mfcom_shifter_parameter_struct* init_struct);
Function descriptions	initialize mfcom_shifter_parameter_struct with the default values
Precondition	-
The called function	-
Input parameter{in}	
init_struct	Mfcom shifter parameter struct, the structure members can refer to members of the structure Struct mfcom_shifter_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize mfcom_shifter_parameter_struct with the default values*/
```

```
mfcom_shifter_parameter_struct mfcom_shifter_struct
```

```
mfcom_shifter_struct_para_init(&mfcom_shifter_struct);
```

mfcom_timer_init

The description of mfcom_timer_init is shown as below:

Table 3-461. Function mfcom_timer_init

Function name	mfcom_timer_init
Function prototype	void mfcom_timer_init(uint32_t timernum, mfcom_timer_parameter_struct* init_struct);
Function descriptions	initialize MFCOM timer parameter
Precondition	-
The called function	-

Input parameter{in}	
timernum	MFCOM timer number
<i>MFCOM_TIMER_x</i>	x = 0...3
Input parameter{in}	
init_struct	Mfcom timer parameter struct, the structure members can refer to members of the structure Struct mfcom timer parameter struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize MFCOM timer parameter */

mfcom_timer_parameter_struct init_struct;

init_struct.trigger_select      = MFCOM_TIMER_TRGSEL_PIN0;
init_struct.trigger_polarity   = MFCOM_TIMER_TRGPOL_ACTIVE_HIGH;
init_struct.pin_config         = MFCOM_TIMER_PINCFG_INPUT;
init_struct.pin_select         = MFCOM_TIMER_PINSEL_PIN0;
init_struct.pin_polarity       = MFCOM_TIMER_PINPOL_ACTIVE_HIGH;
init_struct.mode               = MFCOM_TIMER_BAUDMODE;
init_struct.output             = MFCOM_TIMER_OUT_HIGH_EN_RESET;
init_struct.decrement          = MFCOM_TIMER_DEC_CLK_SHIFT_OUT;
init_struct.reset              = MFCOM_TIMER_RESET_TRIG_TIMEOUT;
init_struct.disable            = MFCOM_TIMER_DISMODE_PINBOTH;
init_struct.enable             = MFCOM_TIMER_ENMODE_TRIGHIGH;
init_struct.stopbit            = MFCOM_TIMER_STOPBIT_TIMDIS;
init_struct.startbit           = MFCOM_TIMER_STARTBIT_ENABLE;

mfcom_timer_init(MFCOM_TIMER_0, &init_struct);

```

mfcom_shifter_init

The description of mfcom_shifter_init is shown as below:

Table 3-462. Function mfcom_shifter_init

Function name	mfcom_shifter_init
Function prototype	void mfcom_shifter_init(uint32_t shifternum,

	mfcom_shifter_parameter_struct* init_struct);
Function descriptions	initialize MFCOM shifter parameter
Precondition	-
The called function	-
Input parameter{in}	
shifternum	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Input parameter{in}	
init_struct	Mfcom shifter parameter struct, the structure members can refer to members of the structure Struct mfcom_shifter_parameter_struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize MFCOM shifter parameter */
```

```
mfcom_shifter_parameter_struct init_struct;
```

```
init_struct.timer_select = MFCOM_SHIFTER_TIMER0;
```

```
init_struct.timer_polarity = MFCOM_SHIFTER_TIMPOL_ACTIVE_HIGH;
```

```
init_struct.pin_config = MFCOM_SHIFTER_PINCFG_INPUT;
```

```
init_struct.pin_select = MFCOM_SHIFTER_PINSEL_PIN0;
```

```
init_struct.pin_polarity = MFCOM_SHIFTER_PINPOL_ACTIVE_HIGH;
```

```
init_struct.mode = MFCOM_SHIFTER_TRANSMIT;
```

```
init_struct.input_source = MFCOM_SHIFTER_INSRC_PIN;
```

```
init_struct.stopbit = MFCOM_SHIFTER_STOPBIT_HIGH;
```

```
init_struct.startbit = MFCOM_SHIFTER_STARTBIT_LOW;
```

```
mfcom_timer_init(MFCOM_SHIFTER_0, &init_struct);
```

mfcom_timer_pin_config

The description of mfcom_timer_pin_config is shown as below:

Table 3-463. Function mfcom_timer_pin_config

Function name	mfcom_timer_pin_config
Function prototype	void mfcom_timer_pin_config(uint32_t timernum, uint32_t mode);
Function descriptions	configure timer pin mode
Precondition	-

The called function	-
Input parameter{in}	
timernum	MFCOM timer number
MFCOM_TIMER_x	x = 0...3
Input parameter{in}	
mode	Output mode
MFCOM_TIMER_PINC FG_INPUT	pin input
MFCOM_TIMER_PINC FG_OPENDRAIN	pin open drain or bidirectional output enable
MFCOM_TIMER_PINC FG_BIDI	Timer cascade pin input/output
MFCOM_TIMER_PINC FG_OUTPUT	pin output
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure timer pin mode */
```

```
mfcom_timer_pin_config(MFCOM_TIMER_0, MFCOM_TIMER_PINCFG_OPENDRAIN);
```

mfcom_shifter_pin_config

The description of mfcom_shifter_pin_config is shown as below:

Table 3-464. Function mfcom_shifter_pin_config

Function name	mfcom_shifter_pin_config
Function prototype	void mfcom_shifter_pin_config(uint32_t shifternum, uint32_t mode);
Function descriptions	configure shifter pin mode
Precondition	-
The called function	-
Input parameter{in}	
shifternum	MFCOM shifter number
MFCOM_TIMER_x	x = 0...3
Input parameter{in}	
mode	Output mode
MFCOM_SHIFTER_PI NCFG_INPUT	pin input
MFCOM_SHIFTER_PI NCFG_OPENDRAIN	pin open drain
MFCOM_SHIFTER_PI	Shifter cascade pin input/output data

<i>NCFG_BIDI</i>	
<i>MFCOM_SHIFTER_PIN</i> <i>NCFG_OUTPUT</i>	pin output
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure shifter pin mode */
mfcom_shifter_pin_config(MFCOM_SHIFTER_0, MFCOM_SHIFTER_PINCFG_BIDI);
```

mfcom_timer_enable

The description of mfcom_timer_enable is shown as below:

Table 3-465. Function mfcom_timer_enable

Function name	mfcom_timer_enable
Function prototype	void mfcom_timer_enable(uint32_t timernum, uint32_t timermode);
Function descriptions	enable MFCOM timer in specific mode
Precondition	-
The called function	-
Input parameter{in}	
timernum	MFCOM timer number
<i>MFCOM_TIMER_x</i>	x = 0...3
Input parameter{in}	
timermode	Timer work mode
<i>MFCOM_TIMER_DISABLE</i>	timer disabled
<i>MFCOM_TIMER_BAUDMODE</i>	dual 8-bit counters baud/bit mode
<i>MFCOM_TIMER_PWM_MODE</i>	dual 8-bit counters PWM mode
<i>MFCOM_TIMER_16BIT_COUNTER</i>	single 16-bit counter mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MFCOM timer in specific mode */
mfcom_timer_enable(MFCOM_TIMER_0, MFCOM_TIMER_BAUDMODE);
```

mfcom_shifter_enable

The description of mfcom_shifter_enable is shown as below:

Table 3-466. Function mfcom_shifter_enable

Function name	mfcom_shifter_enable
Function prototype	void mfcom_shifter_enable(uint32_t shifternum, uint32_t shiftermode);
Function descriptions	enable MFCOM shifter in specific mode
Precondition	-
The called function	-
Input parameter{in}	
shifternum	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Input parameter{in}	
shiftermode	Shifter work mode
<i>MFCOM_SHIFTER_DISABLE</i>	shifter is disabled
<i>MFCOM_SHIFTER_RECEIVE</i>	receive mode
<i>MFCOM_SHIFTER_TRANSMIT</i>	transmit mode
<i>MFCOM_SHIFTER_MATCH_STORE</i>	match store mode
<i>MFCOM_SHIFTER_MATCH_CONTINUOUS</i>	match continuous mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MFCOM shifter in specific mode */
```

```
mfcom_shifter_enable(MFCOM_SHIFTER_0, MFCOM_SHIFTER_RECEIVE);
```

mfcom_timer_disable

The description of mfcom_timer_disable is shown as below:

Table 3-467. Function mfcom_timer_disable

Function name	mfcom_timer_disable
Function prototype	void mfcom_timer_disable(uint32_t timernum);
Function descriptions	disable MFCOM timer
Precondition	-
The called function	-

Input parameter{in}	
timernum	MFCOM timer number
<i>MFCOM_TIMER_x</i>	<i>x = 0...3</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable MFCOM timer */
mfcom_timer_disable(MFCOM_TIMER_0);
```

mfcom_shifter_disable

The description of mfcom_shifter_disable is shown as below:

Table 3-468. Function mfcom_shifter_disable

Function name	mfcom_shifter_disable
Function prototype	void mfcom_shifter_disable(uint32_t shifternum);
Function descriptions	disable MFCOM shifter
Precondition	-
The called function	-
Input parameter{in}	
shifternum	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	<i>x = 0...3</i>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable MFCOM shifter */
mfcom_shifter_disable(MFCOM_SHIFTER_0);
```

mfcom_timer_cmpvalue_set

The description of mfcom_timer_cmpvalue_set is shown as below:

Table 3-469. Function mfcom_timer_cmpvalue_set

Function name	mfcom_timer_cmpvalue_set
Function prototype	void mfcom_timer_cmpvalue_set(uint32_t timernum, uint32_t compare);
Function descriptions	set MFCOM timer compare value
Precondition	-

The called function	-
Input parameter{in}	
timernum	MFCOM timer number
MFCOM_TIMER_x	x = 0...3
Input parameter{in}	
compare	compare value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set MFCOM timer compare value */
mfcom_timer_cmpvalue_set(MFCOM_TIMER_0, 0x0A0A);
```

mfcom_timer_cmpvalue_get

The description of mfcom_timer_cmpvalue_get is shown as below:

Table 3-470. Function mfcom_timer_cmpvalue_get

Function name	mfcom_timer_cmpvalue_get
Function prototype	uint32_t mfcom_timer_cmpvalue_get(uint32_t timernum);
Function descriptions	get MFCOM timer compare value
Precondition	-
The called function	-
Input parameter{in}	
timernum	MFCOM timer number
MFCOM_TIMER_x	x = 0...3
Output parameter{out}	
-	-
Return value	
uint32_t	cmpvalue

Example:

```
/* get MFCOM timer compare value */
uint32_t value = 0;
value = mfcom_timer_cmpvalue_get(MFCOM_TIMER_0);
```

mfcom_timer_dismode_set

The description of mfcom_timer_dismode_set is shown as below:

Table 3-471. Function `mfcom_timer_dismode_set`

Function name	<code>mfcom_timer_dismode_set</code>
Function prototype	<code>void mfcom_timer_dismode_set(uint32_t timernum, uint32_t dismode);</code>
Function descriptions	set MFCOM timer disable mode
Precondition	-
The called function	-
Input parameter{in}	
timernum	MFCOM timer number
<code>MFCOM_TIMER_x</code>	<code>x = 0...3</code>
Input parameter{in}	
dismode	configure conditions that can disable timers and stop decrement
<code>MFCOM_TIMER_DISM</code> <code>ODE_NEVER</code>	timer never disabled
<code>MFCOM_TIMER_DISM</code> <code>ODE_PRE_TMDIS</code>	timer disabled on timer x-1 disable
<code>MFCOM_TIMER_DISM</code> <code>ODE_COMPARE</code>	timer disabled on timer compare
<code>MFCOM_TIMER_DISM</code> <code>ODE_COMPARE_TRIG</code> <code>LOW</code>	timer disabled on timer compare and trigger Low
<code>MFCOM_TIMER_DISM</code> <code>ODE_PINBOTH</code>	timer disabled on pin rising or falling edge
<code>MFCOM_TIMER_DISM</code> <code>ODE_PINBOTH_TRIG</code> <code>HIGH</code>	timer disabled on pin rising or falling edge provided trigger is high
<code>MFCOM_TIMER_DISM</code> <code>ODE_TRIGFALLING</code>	timer disabled on trigger falling edge
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set MFCOM timer disable mode */
```

```
mfcom_timer_dismode_set(MFCOM_TIMER_0, MFCOM_TIMER_DISMODE_COMPARE);
```

`mfcom_shifter_stopbit_set`

The description of `mfcom_shifter_stopbit_set` is shown as below:

Table 3-472. Function `mfcom_shifter_stopbit_set`

Function name	<code>mfcom_shifter_stopbit_set</code>
Function prototype	<code>void mfcom_shifter_stopbit_set(uint32_t shifternum, uint32_t stopbit);</code>

Function descriptions	set MFCOM shifter stopbit
Precondition	-
The called function	-
Input parameter{in}	
shifternum	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Input parameter{in}	
stopbit	stopbit
<i>MFCOM_SHIFTER_STOPBIT_DISABLE</i>	disable shifter stop bit
<i>MFCOM_SHIFTER_STOPBIT_LOW</i>	set shifter stop bit to logic low level
<i>MFCOM_SHIFTER_STOPBIT_HIGH</i>	set shifter stop bit to logic high level
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set MFCOM shifter stopbit */
```

```
mfcom_shifter_stopbit_set(MFCOM_SHIFTER_0, MFCOM_SHIFTER_STOPBIT_LOW);
```

mfcom_buffer_write

The description of mfcom_buffer_write is shown as below:

Table 3-473. Function mfcom_buffer_write

Function name	mfcom_buffer_write
Function prototype	void mfcom_buffer_write(uint32_t shifternum, uint32_t data, uint32_t rwmode);
Function descriptions	write MFCOM shift buffer
Precondition	-
The called function	-
Input parameter{in}	
shifternum	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Input parameter{in}	
data	32-bit data
Input parameter{in}	
rwmode	MFCOM read write mode
<i>MFCOM_RWMODE_NORMAL</i>	read and write in normal mode

<i>MFCOM_RWMODE_BITSWAP</i>	read and write in bit swapped mode
<i>MFCOM_RWMODE_BYTESWAP</i>	read and write in byte swapped mode
<i>MFCOM_RWMODE_BITBYTESWAP</i>	read and write in bit byte swapped mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write MFCOM shift buffer */
```

```
mfcom_buffer_write(MFCOM_SHIFTER_0, 0x6699, MFCOM_RWMODE_BITSWAP);
```

mfcom_buffer_read

The description of mfcom_buffer_read is shown as below:

Table 3-474. Function mfcom_buffer_read

Function name	mfcom_buffer_read
Function prototype	uint32_t mfcom_buffer_read(uint32_t shifternum, uint32_t rwmode);
Function descriptions	read MFCOM shift buffer
Precondition	-
The called function	-
Input parameter{in}	
shifternum	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Input parameter{in}	
rwmode	MFCOM read write mode
<i>MFCOM_RWMODE_NORMAL</i>	read and write in normal mode
<i>MFCOM_RWMODE_BITSWAP</i>	read and write in bit swapped mode
<i>MFCOM_RWMODE_BYTESWAP</i>	read and write in byte swapped mode
<i>MFCOM_RWMODE_BITBYTESWAP</i>	read and write in bit byte swapped mode
Output parameter{out}	
-	-
Return value	
data	32-bit data

Example:

```
/* read MFCOM shift buffer */
```

```
uint32_t data = 0;
```

```
data = mfcom_buffer_read(MFCOM_SHIFTER_0, MFCOM_RWMODE_NORMAL);
```

mfcom_shifter_flag_get

The description of mfcom_shifter_flag_get is shown as below:

Table 3-475. Function mfcom_shifter_flag_get

Function name	mfcom_shifter_flag_get
Function prototype	FlagStatus mfcom_shifter_flag_get(uint32_t shifter);
Function descriptions	get MFCOM shifter flag
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get MFCOM shifter flag */
```

```
flag = mfcom_shifter_flag_get(MFCOM_SHIFTER_0);
```

mfcom_shifter_error_flag_get

The description of mfcom_shifter_error_flag_get is shown as below:

Table 3-476. Function mfcom_shifter_error_flag_get

Function name	mfcom_shifter_error_flag_get
Function prototype	FlagStatus mfcom_shifter_error_flag_get(uint32_t shifter);
Function descriptions	get MFCOM shifter error flag
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get MFCOM shifter error flag */

flag = mfcom_shifter_error_flag_get(MFCOM_SHIFTER_0);
```

mfcom_timer_flag_get

The description of mfcom_timer_flag_get is shown as below:

Table 3-477. Function mfcom_timer_flag_get

Function name	mfcom_timer_flag_get
Function prototype	FlagStatus mfcom_timer_flag_get(uint32_t timer);
Function descriptions	get MFCOM timer flag
Precondition	-
The called function	-
Input parameter{in}	
timer	MFCOM timer number
<i>MFCOM_TIMER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get MFCOM timer flag */

flag = mfcom_timer_flag_get(MFCOM_TIMER_0);
```

mfcom_shifter_interrupt_flag_get

The description of mfcom_shifter_interrupt_flag_get is shown as below:

Table 3-478. Function mfcom_shifter_interrupt_flag_get

Function name	mfcom_shifter_interrupt_flag_get
Function prototype	FlagStatus mfcom_shifter_interrupt_flag_get(uint32_t shifter);
Function descriptions	get MFCOM shifter interrupt flag
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get MFCOM shifter interrupt flag */

flag = mfcom_shifter_interrupt_flag_get(MFCOM_SHIFTER_0);
```

mfcom_shifter_error_interrupt_flag_get

The description of mfcom_shifter_error_interrupt_flag_get is shown as below:

Table 3-479. Function mfcom_shifter_error_interrupt_flag_get

Function name	mfcom_shifter_error_interrupt_flag_get
Function prototype	FlagStatus mfcom_shifter_error_interrupt_flag_get(uint32_t shifter);
Function descriptions	get MFCOM shifter error interrupt flag
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get MFCOM shifter error interrupt flag */

flag = mfcom_shifter_error_interrupt_flag_get (MFCOM_SHIFTER_0);
```

mfcom_timer_interrupt_flag_get

The description of mfcom_enable is shown as below:

Table 3-480. Function mfcom_timer_interrupt_flag_get

Function name	mfcom_timer_interrupt_flag_get
Function prototype	FlagStatus mfcom_timer_interrupt_flag_get(uint32_t timer);
Function descriptions	get MFCOM timer interrupt flag
Precondition	-
The called function	-
Input parameter{in}	
timer	MFCOM timer number
<i>MFCOM_TIMER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get MFCOM timer interrupt flag */

flag = mfcom_timer_interrupt_flag_get (MFCOM_TIMER_0);
```

mfcom_shifter_flag_clear

The description of mfcom_shifter_flag_clear is shown as below:

Table 3-481. Function mfcom_shifter_flag_clear

Function name	mfcom_shifter_flag_clear
Function prototype	void mfcom_shifter_flag_clear(uint32_t shifter);
Function descriptions	clear MFCOM shifter flag
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear MFCOM shifter flag */

flag = mfcom_shifter_flag_clear(MFCOM_SHIFTER_0);
```

mfcom_shifter_error_flag_clear

The description of mfcom_shifter_error_flag_clear is shown as below:

Table 3-482. Function mfcom_shifter_error_flag_clear

Function name	mfcom_shifter_error_flag_clear
Function prototype	void mfcom_shifter_error_flag_clear (uint32_t shifter);
Function descriptions	clear MFCOM shifter error flag
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear MFCOM shifter flag */
```

```
flag = mfcom_shifter_flag_clear(MFCOM_SHIFTER_0);
```

mfcom_timer_flag_clear

The description of mfcom_timer_flag_clear is shown as below:

Table 3-483. Function mfcom_timer_flag_clear

Function name	mfcom_timer_flag_clear
Function prototype	void mfcom_timer_flag_clear(uint32_t timer);
Function descriptions	clear MFCOM timer flag
Precondition	-
The called function	-
Input parameter{in}	
timer	MFCOM timer number
<i>MFCOM_TIMER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear MFCOM timer flag */
```

```
mfcom_timer_flag_clear (MFCOM_TIMER_0);
```

mfcom_shifter_interrupt_enable

The description of mfcom_shifter_interrupt_enable is shown as below:

Table 3-484. Function mfcom_shifter_interrupt_enable

Function name	mfcom_shifter_interrupt_enable
Function prototype	void mfcom_shifter_interrupt_enable (uint32_t shifter);
Function descriptions	enable MFCOM shifter interrupt
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MFCOM shifter interrupt */

mfcom_shifter_interrupt_enable (MFCOM_SHIFTER_0);
```

mfcom_shifter_error_interrupt_enable

The description of mfcom_shifter_error_interrupt_enable is shown as below:

Table 3-485. Function mfcom_shifter_error_interrupt_enable

Function name	mfcom_shifter_error_interrupt_enable
Function prototype	void mfcom_shifter_error_interrupt_enable (uint32_t shifter);
Function descriptions	enable MFCOM shifter error interrupt
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MFCOM shifter error interrupt */

mfcom_shifter_error_interrupt_enable (MFCOM_SHIFTER_0);
```

mfcom_timer_interrupt_enable

The description of mfcom_timer_interrupt_enable is shown as below:

Table 3-486. Function mfcom_timer_interrupt_enable

Function name	mfcom_timer_interrupt_enable
Function prototype	void mfcom_timer_interrupt_enable (uint32_t timer);
Function descriptions	enable MFCOM timer interrupt
Precondition	-
The called function	-
Input parameter{in}	
timer	MFCOM timer number
<i>MFCOM_TIMER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MFCOM timer interrupt */
```

```
mfcom_timer_interrupt_enable (MFCOM_TIMER_0);
```

mfcom_shifter_dma_enable

The description of mfcom_shifter_dma_enable is shown as below:

Table 3-487. Function mfcom_shifter_dma_enable

Function name	mfcom_shifter_dma_enable
Function prototype	void mfcom_shifter_dma_enable (uint32_t shifter);
Function descriptions	enable MFCOM shifter dma
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
MFCOM_SHIFTER_x	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MFCOM shifter dma */
```

```
mfcom_shifter_dma_enable (MFCOM_SHIFTER_0);
```

mfcom_shifter_interrupt_disable

The description of mfcom_shifter_interrupt_disable is shown as below:

Table 3-488. Function mfcom_shifter_interrupt_disable

Function name	mfcom_shifter_interrupt_disable
Function prototype	void mfcom_shifter_interrupt_disable (uint32_t shifter);
Function descriptions	disable MFCOM shifter interrupt
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
MFCOM_SHIFTER_x	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable MFCOM shifter interrupt */

mfcom_shifter_interrupt_disable (MFCOM_SHIFTER_0);
```

mfcom_shifter_error_interrupt_disable

The description of mfcom_shifter_error_interrupt_disable is shown as below:

Table 3-489. Function mfcom_shifter_error_interrupt_disable

Function name	mfcom_shifter_error_interrupt_disable
Function prototype	void mfcom_shifter_error_interrupt_disable (uint32_t shifter);
Function descriptions	disable MFCOM shifter error interrupt
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
<i>MFCOM_SHIFTER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable MFCOM shifter error interrupt */

mfcom_shifter_error_interrupt_disable (MFCOM_SHIFTER_0);
```

mfcom_timer_interrupt_disable

The description of mfcom_timer_interrupt_disable is shown as below:

Table 3-490. Function mfcom_timer_interrupt_disable

Function name	mfcom_timer_interrupt_disable
Function prototype	void mfcom_timer_interrupt_disable (uint32_t timer);
Function descriptions	disable MFCOM timer interrupt
Precondition	-
The called function	-
Input parameter{in}	
timer	MFCOM timer number
<i>MFCOM_TIMER_x</i>	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable MFCOM timer interrupt */

mfcom_timer_interrupt_disable (MFCOM_TIMER_0);
```

mfcom_shifter_dma_disable

The description of mfcom_shifter_dma_disable is shown as below:

Table 3-491. Function mfcom_shifter_dma_disable

Function name	mfcom_shifter_dma_disable
Function prototype	void mfcom_shifter_dma_disable (uint32_t shifter);
Function descriptions	disable MFCOM shifter dma
Precondition	-
The called function	-
Input parameter{in}	
shifter	MFCOM shifter number
MFCOM_SHIFTER_x	x = 0...3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable MFCOM shifter dma */

mfcom_shifter_dma_disable (MFCOM_SHIFTER_0);
```

3.16. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick. The NVIC and SysTick registers are listed in chapter [3.16.1](#), the MISC firmware functions are introduced in chapter [3.16.2](#).

3.16.1. Descriptions of Peripheral registers

Table 3-492. 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

Registers	Descriptions
IPR ⁽¹⁾	Interrupt Priority Register
STIR ⁽¹⁾	Software Trigger Interrupt Register
CPUID ⁽²⁾	CPUID Base Register
ICSR ⁽²⁾	Interrupt Control and State Register
VTOR ⁽²⁾	Vector Table Offset Register
AIRCR ⁽²⁾	Application Interrupt and Reset Control Register
SCR ⁽²⁾	System Control Register
CCR ⁽²⁾	Configuration Control Register
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_cm33h file

Table 3-493. 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

3.16.2. Descriptions of Peripheral functions

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

Table 3-494. 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 base address
system_lowpower_set	set the state of the low power mode
system_lowpower_reset	reset the state of the low power mode
systick_clksource_set	set the systick clock source

Enum IRQn_Type

Table 3-495. IRQn_Type

Member name	Function description
WWDGT_IRQn	window watchDog timer interrupt
LVD_IRQn	LVD through EXTI line detect interrupt
RTC_IRQn	RTC global interrupt

Member name	Function description
FMC_IRQn	FMC global interrupt
RCU_IRQn	RCU global interrupt
EXTI0_IRQn	EXTI line 0 interrupts
EXTI1_IRQn	EXTI line 1 interrupts
EXTI2_IRQn	EXTI line 2 interrupts
EXTI3_IRQn	EXTI line 3 interrupts
EXTI4_IRQn	EXTI line 4 interrupts
DMA0_Channel0_IRQn	DMA0 channel 0 global interrupt
DMA0_Channel1_IRQn	DMA0 channel 1 global interrupt
DMA0_Channel2_IRQn	DMA0 channel 2 global interrupt
DMA0_Channel3_IRQn	DMA0 channel 3 global interrupt
DMA0_Channel4_IRQn	DMA0 channel 4 global interrupt
DMA0_Channel5_IRQn	DMA0 channel 5 global interrupt
DMA0_Channel6_IRQn	DMA0 channel 6 global interrupt
ADC0_1_IRQn	ADC0 and ADC1 interrupt
CAN0_Message_IRQn	CAN0 interrupt for message buffer
CAN0_Busoff_IRQn	CAN0 interrupt for Bus off / Bus off done
CAN0_Error_IRQn	CAN0 interrupt for Error
CAN0_FastError_IRQn	CAN0 interrupt for error in fast transmission
CAN0_TEC_IRQn	CAN0 interrupt for transmit warning
CAN0_REC_IRQn	CAN0 interrupt for receive warning
CAN0_WKUP_IRQn	CAN0 wakeup through EXTI Line detection interrupt
TIMER0_BRK_UP_TRG_CMT_IRQn	TIMER0 Break, update, trigger and commutation interrupt
TIMER0_Channel_IRQn	TIMER0 Capture Compare interrupt
TIMER1_IRQn	TIMER1 global interrupt
TIMER19_BRK_UP_TRG_CMT_IRQn	TIMER19 Break, update, trigger and commutation interrupt
TIMER19_Channel_IRQn	TIMER19 Capture Compare interrupt
I2C0_EV_IRQn	I2C0 event interrupt
I2C0_ER_IRQn	I2C0 error interrupt
I2C1_EV_IRQn	I2C1 event interrupt
I2C1_ER_IRQn	I2C1 error interrupt
SPI0_IRQn	SPI0 global interrupt
SPI1_IRQn	SPI1 global interrupt
USART0_IRQn	USART0 global interrupt
USART1_IRQn	USART1 global interrupt
USART2_IRQn	USART2 global interrupt
EXTI10_15_IRQn	EXTI Line10-15 interrupt
EXTI5_9_IRQn	EXTI Line5-9 interrupt
TAMPER_IRQn	BKP Tamper interrupt
TIMER20_BRK_UP_TRG_CMT_IRQn	TIMER20 Break, update, trigger and commutation interrupt
TIMER20_Channel_IRQn	TIMER20 Capture Compare interrupt

Member name	Function description
TIMER7_BRK_UP_TRG_CMT_IRQn	TIMER7 Break, update, trigger and commutation interrupt
TIMER7_Channel_IRQn	TIMER7 Capture Compare interrupt
DMAMUX_IRQn	DMA MUX interrupt
SRAMC_ECCSE_IRQn	SYSCFG SRAM ECC single err interrupt
CMP_IRQn	CMP through EXTI Line detection interrupt
OVD_IRQn	Over voltage detector through EXTI Line detection interrupt
TIMER5_DAC_IRQn	TIMER5 interrupt, DAC global interrupt
TIMER6_IRQn	TIMER6 global interrupt
DMA1_Channel0_IRQn	DMA1 Channel 0 global interrupt
DMA1_Channel1_IRQn	DMA1 Channel 1 global interrupt
DMA1_Channel2_IRQn	DMA1 Channel 2 global interrupt
DMA1_Channel3_IRQn	DMA1 Channel 3 global interrupt
DMA1_Channel4_IRQn	DMA1 Channel 4 global interrupt
CAN1_WKUP_IRQn	CAN1 wakeup through EXTI Line detection interrupt
CAN1_Message_IRQn	CAN1 interrupt for message buffer
CAN1_Busoff_IRQn	CAN1 interrupt for Bus off / Bus off done
CAN1_Error_IRQn	CAN1 interrupt for error
CAN1_FastError_IRQn	CAN1 interrupt for error in fast transmission
CAN1_TEC_IRQn	CAN1 interrupt for transmit warning
CAN1_REC_IRQn	CAN1 interrupt for receive warning
FPU_IRQn	FPU global interrupt
MFCOM_IRQn	MFCOM interrupt

nvic_priority_group_set

The description of nvic_priority_group_set is shown as below:

Table 3-496. Function nvic_priority_group_set

Function name	nvic_priority_group_set
Function prototype	void nvic_priority_group_set(uint32_t nvic_prigroup);
Function descriptions	set the priority group
Precondition	-
The called functions	-
Input parameter{in}	
nvic_prigroup	priority group
NVIC_PRIGROUP_PR E0_SUB4	0 bits for pre-emption priority 4 bits for subpriority
NVIC_PRIGROUP_PR E1_SUB3	1 bits for pre-emption priority 3 bits for subpriority
NVIC_PRIGROUP_PR E2_SUB2	2 bits for pre-emption priority 2 bits for subpriority
NVIC_PRIGROUP_PR	3 bits for pre-emption priority 1 bits for subpriority

<i>E3_SUB1</i>	
<i>NVIC_PRIGROUP_PR</i> <i>E4_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 3-497. Function nvic_irq_enable

Function name	nvic_irq_enable
Function prototype	void nvic_irq_enable(uint8_t nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t nvic_irq_sub_priority);
Function descriptions	enable NVIC interrupt request
Precondition	-
The called functions	nvic_priority_group_set
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to Table 3-495. IRQn_Type
Input parameter{in}	
nvic_irq_pre_priority	the pre-emption priority needed to set
Input parameter{in}	
nvic_irq_sub_priority	the subpriority needed to set
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

nvic_irq_disable

The description of nvic_irq_disable is shown as below:

Table 3-498. Function nvic_irq_disable

Function name	nvic_irq_disable
----------------------	------------------

Function prototype	void nvic_irq_disable(uint8_t nvic_irq);
Function descriptions	disable NVIC interrupt request
Precondition	-
The called functions	-
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to enum Table 3-495. 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 3-499. 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	NVIC_SystemReset
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the MCU */
nvic_system_reset();
```

nvic_vector_table_set

The description of nvic_vector_table_set is shown as below:

Table 3-500. 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 base 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>H</i>	
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 3-501. Function system_lowpower_set

Function name	system_lowpower_set
Function prototype	void system_lowpower_set(uint8_t lowpower_mode);
Function descriptions	set the state of the low power mode
Precondition	-
The called functions	-
Input parameter{in}	
lowpower_mode	the low power mode state
<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 3-502. Function system_lowpower_reset

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

Example:

```
/* the system will exit low power mode by exiting from ISR */
system_lowpower_reset(SCB_LPM_SLEEP_EXIT_ISR);
```

systick_clksource_set

The description of systick_clksource_set is shown as below:

Table 3-503. 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);
```

3.17. PMU

According to the Power management unit (PMU), provides three types of power saving modes, including Sleep, Deep-sleep, Standby mode. The PMU registers are listed in chapter [3.17.1](#), the PMU firmware functions are introduced in chapter [3.17.2](#).

3.17.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

Table 3-504. PMU Registers

Registers	Descriptions
PMU_CTL	PMU control register
PMU_CS	PMU control and status register

3.17.2. Descriptions of Peripheral functions

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

Table 3-505. PMU firmware function

Function name	Function description
pmu_deinit	reset PMU registers
pmu_lvd_select	select low voltage detector threshold
pmu_lvd_disable	disable PMU lvd
pmu_ovd_select	select over voltage detector threshold
pmu_ovd_disable	disable PMU ovd
pmu_lowdriver_mode_enable	enable low-driver mode in deep-sleep mode
pmu_lowdriver_mode_disable	disable low-driver mode in deep-sleep mode
pmu_sram1_poweroff_mode_enable	SRAM1 power off in deep-sleep mode
pmu_sram1_poweroff_mode_disable	SRAM1 power on in deep-sleep mode
pmu_sram2_poweroff_mode_enable	SRAM2 power off in deep-sleep mode
pmu_sram2_poweroff_mode_disable	SRAM2 power on in deep-sleep mode
pmu_to_sleepmode	PMU work in sleep mode
pmu_to_deepsleepmode	PMU work in deepsleep mode
pmu_to_standbymode	PMU work in standby mode

Function name	Function description
pmu_wakeup_pin_enable	enable PMU wakeup pin
pmu_wakeup_pin_disable	disable PMU wakeup pin
pmu_backup_write_enable	enable write access to the registers in backup domain
pmu_backup_write_disable	disable write access to the registers in backup domain
pmu_flag_get	get flag state
pmu_flag_clear	clear flag bit

pmu_deinit

The description of pmu_deinit is shown as below:

Table 3-506. Function pmu_deinit

Function name	pmu_deinit
Function prototype	void pmu_deinit(void);
Function descriptions	reset PMU registers
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 3-507. 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
PMU_LVDT_0	voltage threshold is 2.9V
PMU_LVDT_1	voltage threshold is 3.1V
PMU_LVDT_2	voltage threshold is 3.3V

<i>PMU_LVDT_3</i>	voltage threshold is 3.5V
<i>PMU_LVDT_4</i>	voltage threshold is 4.0V
<i>PMU_LVDT_5</i>	voltage threshold is 4.2V
<i>PMU_LVDT_6</i>	voltage threshold is 4.4V
<i>PMU_LVDT_7</i>	voltage threshold is 4.6V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select low voltage detector threshold as 4.6V */
```

```
pmu_lvd_select(PMU_LVDT_7);
```

pmu_lvd_disable

The description of pmu_lvd_disable is shown as below:

Table 3-508. 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_ovd_select

The description of pmu_ovd_select is shown as below:

Table 3-509. Function pmu_ovd_select

Function name	pmu_ovd_select
Function prototype	void pmu_ovd_select(uint32_t ovd_t_n);
Function descriptions	select over voltage detector threshold

Precondition	-
The called functions	-
Input parameter{in}	
lvdt_n	voltage threshold value
<i>PMU_OVDT_0</i>	voltage threshold is 5.0V
<i>PMU_OVDT_1</i>	voltage threshold is 5.5V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select over voltage detector threshold as 5.5V */
```

```
pmu_vd_select(PMU_OVDT_1);
```

pmu_ovd_disable

The description of pmu_ovd_disable is shown as below:

Table 3-510. Function pmu_ovd_disable

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

Example:

```
/* disable PMU ovd */
```

```
pmu_ovd_disable();
```

pmu_lowdriver_mode_enable

The description of pmu_lowdriver_mode_enable is shown as below:

Table 3-511. Function pmu_lowdriver_mode_enable

Function name	pmu_lowdriver_mode_enable
Function prototype	void pmu_lowdriver_mode_enable(void);

Function descriptions	enable low-driver mode in deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable low-driver mode in deep-sleep mode */
```

```
pmu_lowdriver_mode_enable ();
```

pmu_lowdriver_mode_disable

The description of pmu_lowdriver_mode_disable is shown as below:

Table 3-512. Function pmu_lowdriver_mode_disable

Function name	pmu_lowdriver_mode_disable
Function prototype	void pmu_lowdriver_mode_disable(void);
Function descriptions	disable low-driver mode in deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable low-driver mode in deep-sleep mode */
```

```
pmu_lowdriver_mode_disable();
```

pmu_sram1_poweroff_mode_enable

The description of pmu_sram1_poweroff_mode_enable is shown as below:

Table 3-513. Function pmu_sram1_poweroff_mode_enable

Function name	pmu_sram1_poweroff_mode_enable
Function prototype	void pmu_sram1_poweroff_mode_enable(void);
Function descriptions	SRAM1 power off in deep-sleep mode

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

Example:

```
/* SRAM1 power off in deep-sleep mode */
```

```
pmu_sram1_poweroff_mode_enable();
```

pmu_sram1_poweroff_mode_disable

The description of pmu_sram1_poweroff_mode_disable is shown as below:

Table 3-514. Function pmu_sram1_poweroff_mode_disable

Function name	pmu_sram1_poweroff_mode_disable
Function prototype	void pmu_sram1_poweroff_mode_disable(void);
Function descriptions	SRAM1 power on in deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SRAM1 power on in deep-sleep mode */
```

```
pmu_sram1_poweroff_mode_disable();
```

pmu_sram2_poweroff_mode_enable

The description of pmu_sram2_poweroff_mode_enable is shown as below:

Table 3-515. Function pmu_sram2_poweroff_mode_enable

Function name	pmu_sram2_poweroff_mode_enable
Function prototype	void pmu_sram2_poweroff_mode_enable(void);
Function descriptions	SRAM2 power off in deep-sleep mode
Precondition	-

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

Example:

```
/* SRAM2 power off in deep-sleep mode */
pmu_sram2_poweroff_mode_enable();
```

pmu_sram2_poweroff_mode_disable

The description of pmu_sram2_poweroff_mode_disable is shown as below:

Table 3-516. Function pmu_sram2_poweroff_mode_disable

Function name	pmu_sram2_poweroff_mode_disable
Function prototype	void pmu_sram2_poweroff_mode_disable(void);
Function descriptions	SRAM2 power on in deep-sleep mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SRAM2 power on in deep-sleep mode */
pmu_sram2_poweroff_mode_disable ( );
```

pmu_to_sleepmode

The description of pmu_to_sleepmode is shown as below:

Table 3-517. Function pmu_to_sleepmode

Function name	pmu_to_sleepmode
Function prototype	void pmu_to_sleepmode(uint8_t sleepmodecmd);
Function descriptions	PMU work in 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 3-518. 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 in 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	low-driver mode
<i>PMU_LOWDRIVER_ENABLE</i>	low-driver mode enable in deep-sleep mode
<i>PMU_LOWDRIVER_DISABLE</i>	low-driver mode disable 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 in deepsleep mode */
```

```
pmu_to_deepsleepmode (PMU_LDO_NORMAL, PMU_LOWDRIVER_DISABLE,
WFI_CMD);
```

pmu_to_standbymode

The description of pmu_to_standbymode is shown as below:

Table 3-519. 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_standby();
```

pmu_wakeup_pin_enable

The description of pmu_wakeup_pin_enable is shown as below:

Table 3-520. 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	Wakeup pin
PMU_WAKEUP_PIN0	WKUP Pin 0 (PA0)
PMU_WAKEUP_PIN1	WKUP Pin 1 (PC13)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable wakeup pin0 */
pmu_wakeup_pin_enable(PMU_WAKEUP_PIN0);
```

pmu_wakeup_pin_disable

The description of pmu_wakeup_pin_disable is shown as below:

Table 3-521. 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	Wakeup pin
PMU_WAKEUP_PIN0	WKUP Pin 0 (PA0)
PMU_WAKEUP_PIN1	WKUP Pin 1 (PC13)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable wakeup pin0 */
pmu_wakeup_pin_disable(PMU_WAKEUP_PIN0);
```

pmu_backup_write_enable

The description of pmu_backup_write_enable is shown as below:

Table 3-522. 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 3-523. Function pmu_backup_write_disable

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

Example:

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

pmu_flag_get

The description of pmu_flag_get is shown as below:

Table 3-524. Function pmu_flag_get

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

<i>PMU_FLAG_WAKEUP</i>	wakeup flag
<i>PMU_FLAG_STANDBY</i>	standby flag
<i>PMU_FLAG_LVD</i>	lvd flag
<i>PMU_FLAG_OVD</i>	ovd flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get flag state */
```

```
FlagStatus status;
```

```
status = pmu_flag_get(PMU_FLAG_WAKEUP);
```

pmu_flag_clear

The description of pmu_flag_clear is shown as below:

Table 3-525. Function pmu_flag_clear

Function name	pmu_flag_clear
Function prototype	void pmu_flag_clear(uint32_t flag);
Function descriptions	clear flag bit
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag
<i>PMU_FLAG_RESET_WAKEUP</i>	reset wakeup flag
<i>PMU_FLAG_RESET_STANDBY</i>	reset standby flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear flag bit */
```

```
pmu_flag_clear(PMU_FLAG_RESET_STANDBY);
```

3.18. 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 [3.18.1](#), the RCU firmware functions are introduced in chapter [3.18.2](#).

3.18.1. Descriptions of Peripheral registers

Table 3-526. RCU Registers

Registers	Descriptions
RCU_CTL	Control register
RCU_CFG0	Clock configuration register 0
RCU_INT	Clock interrupt register
RCU_APB2RST	APB2 reset register
RCU_APB1RST	APB1 reset register
RCU_AHBEN	AHB enable register
RCU_APB2EN	APB2 enable register
RCU_APB1EN	APB1 enable register
RCU_BDCTL	Backup domain control register
RCU_RSTSCK	Reset source/clock register
RCU_AHBRST	AHB reset register
RCU_CFG1	Clock configuration register 1
RCU_CFG2	Clock configuration register 2
RCU_VKEY	Voltage key register
RCU_DSV	Deep-sleep mode voltage register

3.18.2. Descriptions of Peripheral functions

Table 3-527. RCU firmware function

Function name	Function description
rcu_deinit	deinitialize the RCU, reset the value of all RCU registers into initial values
rcu_periph_clock_enable	enable the peripherals clock
rcu_periph_clock_disable	disable the peripherals clock
rcu_periph_reset_enable	enable the peripherals reset
rcu_periph_reset_disable	disable the peripheral reset
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_bkp_reset_enable	enable the BKP domain reset
rcu_bkp_reset_disable	disable the BKP domain reset
rcu_system_clock_source_config	configure the system clock source
rcu_system_clock_source_get	get the system clock source
rcu_ahb_clock_config	configure the AHB clock prescaler selection
rcu_apb1_clock_config	configure the APB1 clock prescaler selection

Function name	Function description
rcu_apb2_clock_config	configure the APB2 clock prescaler selection
rcu_ckout_config	configure the CK_OUT clock source and divider
rcu_pll_config	configure the main PLL clock
rcu_double_pll_enable	enable double PLL clock
rcu_double_pll_disable	disable double PLL clock
rcu_system_reset_enable	enable RCU system reset
rcu_system_reset_disable	disable RCU system reset
rcu_adc_clock_config	configure the ADC clock source and prescaler selection
rcu_rtc_clock_config	configure the RTC clock source selection
rcu_usart_clock_config	configure the usart clock
rcu_can_clock_config	configure the CAN clock source 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_hxtal_frequency_scale_select	configure the HXTAL frequency scale select
rcu_hxtal_prediv_config	configure the HXTAL divider used as input of PLL
rcu_irc8m_adjust_value_set	set the IRC8M adjust value
rcu_hxtal_clock_monitor_enable	enable the HXTAL clock monitor
rcu_hxtal_clock_monitor_disable	disable the HXTAL clock monitor
rcu_lxtal_clock_monitor_enable	enable the LXTAL clock monitor
rcu_lxtal_clock_monitor_disable	disable the LXTAL clock monitor
rcu_pll_clock_monitor_enable	enable the PLL clock monitor
rcu_pll_clock_monitor_disable	disable the PLL 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 and peripheral clock frequency
rcu_flag_get	get the clock stabilization and peripheral reset flags
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_interrupt_enable	enable the stabilization interrupt
rcu_interrupt_disable	disable the stabilization interrupt

Enum rcu_periph_enum

Table 3-528. Enum rcu_periph_enum

enum name	Function description
RCU_DMA0	DMA0 clock

enum name	Function description
RCU_DMA1	DMA1 clock
RCU_DMAMUX	DMAMUX clock
RCU_CRC	CRC clock
RCU_MFCOM	MFCOM clock
RCU_GPIOA	GPIOA clock
RCU_GPIOB	GPIOB clock
RCU_GPIOC	GPIOC clock
RCU_GPIOD	GPIOD clock
RCU_GPIOE	GPIOE clock
RCU_GPIOF	GPIOF clock
RCU_SYSCFG	SYSCFG clock
RCU_CMP	CMP clock
RCU_ADC0	ADC0 clock
RCU_ADC1	ADC1 clock
RCU_TIMER0	TIMER0 clock
RCU_SPI0	SPI0 clock
RCU_TIMER7	TIMER7 clock
RCU_USART0	USART0 clock
RCU_TIMER19	TIMER19 clock
RCU_TIMER20	TIMER20 clock
RCU_TRIGSEL	TRIGSEL clock
RCU_CAN0	CAN0 clock
RCU_CAN1	CAN1 clock
RCU_TIMER1	TIMER1 clock
RCU_TIMER5	TIMER5 clock
RCU_TIMER6	TIMER6 clock
RCU_WWDGT	WWDGT clock
RCU_SPI1	SPI1 clock
RCU_USART1	USART1 clock
RCU_USART2	USART2 clock
RCU_I2C0	I2C0 clock
RCU_I2C1	I2C1 clock
RCU_BKP	BKP clock
RCU_PMU	PMU clock
RCU_DAC	DAC clock
RCU_RTC	RTC clock

Enum rcu_periph_sleep_enum

Table 3-529. Enum rcu_periph_sleep_enum

enum name	Function description
RCU_SRAM_SLP	SRAM clock

enum name	Function description
RCU_FMC_SLP	FMC clock

Enum rcu_periph_reset_enum

Table 3-530. Enum rcu_periph_reset_enum

enum name	Function description
RCU_DMA0RST	DMA0 clock reset
RCU_DMA1RST	DMA1 clock reset
RCU_DMAMUXRST	DMAMUX clock reset
RCU_CRCRST	CRC reset
RCU_MFCOMRST	MFCOM clock reset
RCU_GPIOARST	GPIOA reset
RCU_GPIOBRST	GPIOB reset
RCU_GPIOCRST	GPIOC reset
RCU_GPIODRST	GPIOD reset
RCU_GPIOERST	GPIOE reset
RCU_GPIOFRST	GPIOF reset
RCU_SYSCFGRST	SYSCFG reset
RCU_CMPRST	CMP reset
RCU_ADC0RST	ADC0 reset
RCU_ADC1RST	ADC1 reset
RCU_TIMER0RST	TIMER0 reset
RCU_SPI0RST	SPI0 reset
RCU_TIMER7RST	TIMER7 reset
RCU_USART0RST	USART0 reset
RCU_TIMER19RST	TIMER19 reset
RCU_TIMER20RST	TIMER20 reset
RCU_CAN0RST	CAN0 reset
RCU_CAN1RST	CAN1 reset
RCU_TIMER1RST	TIMER1 reset
RCU_TIMER5RST	TIMER5 reset
RCU_TIMER6RST	TIMER6 reset
RCU_WWDGTRST	WWDGT reset
RCU_SPI1RST	SPI1 reset
RCU_USART1RST	USART1 reset
RCU_USART2RST	USART2 reset
RCU_I2C0RST	I2C0 reset
RCU_I2C1RST	I2C1 reset
RCU_PMURST	PMU reset
RCU_DACRST	DAC reset

Enum rcu_flag_enum

Table 3-531. Enum rcu_flag_enum

enum name	Function description
RCU_FLAG_IRC8MSTB	IRC8M stabilization flag
RCU_FLAG_HXTALSTB	HXTAL stabilization flag
RCU_FLAG_PLLSTB	PLL stabilization flag
RCU_FLAG_LXTALSTB	LXTAL stabilization flag
RCU_FLAG_IRC40KSTB	IRC40K stabilization flag
RCU_FLAG_BORRST	BOR reset flag
RCU_FLAG_LOCKUPRST	CPU LOCK UP error reset flag
RCU_FLAG_LVDRST	low voltage detect error reset flag
RCU_FLAG_ECCRST	2 bits ECC error reset flag
RCU_FLAG_LOHRST	lost of HXTAL error reset flag
RCU_FLAG_LOPRST	lost of PLL error reset flag
RCU_FLAG_V11RST	1.1V domain Power reset flag
RCU_FLAG_OBLRST	option byte loader reset flag
RCU_FLAG_EPRST	external PIN reset flag
RCU_FLAG_PORRST	power reset flag
RCU_FLAG_SWRST	software reset flag
RCU_FLAG_FWDGTRST	FWDGT reset flag
RCU_FLAG_WWDGTRST	WWDGT reset flag
RCU_FLAG_LPRST	LP reset flag

Enum rcu_int_flag_enum

Table 3-532. Enum rcu_int_flag_enum

enum name	Function description
RCU_INT_FLAG_IRC40KSTB	IRC40K stabilization interrupt flag
RCU_INT_FLAG_LXTALSTB	LXTAL stabilization interrupt flag
RCU_INT_FLAG_IRC8MSTB	IRC8M stabilization interrupt flag
RCU_INT_FLAG_HXTALSTB	HXTAL stabilization interrupt flag

enum name	Function description
RCU_INT_FLAG_PLLS TB	PLL stabilization interrupt flag
RCU_INT_FLAG_LCK M	LXTAL clock monitor interrupt flag
RCU_INT_FLAG_PLLM	PLL clock monitor interrupt flag
RCU_INT_FLAG_CKM	HXTAL clock stuck interrupt flag

Enum rcu_int_flag_clear_enum

Table 3-533. Enum rcu_int_flag_clear_enum

enum name	Function description
RCU_INT_FLAG_IRC4 0KSTB_CLR	IRC40K stabilization interrupt flag clear
RCU_INT_FLAG_LXTA LSTB_CLR	LXTAL stabilization interrupt flag clear
RCU_INT_FLAG_IRC8 MSTB_CLR	IRC8M stabilization interrupt flag clear
RCU_INT_FLAG_HXTA LSTB_CLR	HXTAL stabilization interrupt flag clear
RCU_INT_FLAG_PLLS TB_CLR	PLL stabilization interrupt flag clear
RCU_INT_FLAG_LCK M_CLR	LXTAL clock stuck interrupt flag clear
RCU_INT_FLAG_PLLM _CLR	PLL clock monitor interrupt clear
RCU_INT_FLAG_CKM _CLR	CKM interrupt flag clear

Enum rcu_int_enum

Table 3-534. Enum rcu_int_enum

enum name	Function description
RCU_INT_IRC40KSTB	IRC40K stabilization interrupt
RCU_INT_LXTALSTB	LXTAL stabilization interrupt
RCU_INT_IRC8MSTB	IRC8M stabilization interrupt
RCU_INT_HXTALSTB	HXTAL stabilization interrupt
RCU_INT_PLLSTB	PLL stabilization interrupt
RCU_INT_LCKM	LXTAL clock monitor interrupt
RCU_INT_PLLM	PLL clock monitor interrupt

Enum rcu_osci_type_enum

Table 3-535. Enum rcu_osci_type_enum

enum name	Function description
RCU_HXTAL	HXTAL
RCU_LXTAL	LXTAL
RCU_IRC8M	IRC8M
RCU_IRC40K	IRC40K
RCU_PLL_CK	PLL

Enum rcu_clock_freq_enum

Table 3-536. 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_USART1	USART1 clock
CK_USART2	USART2 clock

rcu_deinit

The description of rcu_deinit is shown as below:

Table 3-537. 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	-
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 3-538. 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 3-528. Enum rcu_periph_enum
<i>RCU_GPIOx</i>	GPIOx ports clock (x = A,B,C,D,E,F)
<i>RCU_DMAx</i>	DMAx clock (x = 0, 1)
<i>RCU_CRC</i>	CRC clock
<i>RCU_SYSCFG</i>	SYSCFG clock
<i>RCU_CMP</i>	CMP clock
<i>RCU_ADCx</i>	ADCx clock (x = 0, 1)
<i>RCU_TIMERx</i>	TIMERx clock (x = 0,1,5,6,7,19,20)
<i>RCU_SPIx</i>	SPIx clock (x = 0,1)
<i>RCU_USARTx</i>	USARTx clock (x = 0,1,2)
<i>RCU_MFCOM</i>	MFCOM clock
<i>RCU_TRIGSEL</i>	TRIGSEL clock
<i>RCU_CANx</i>	CANx clock (x = 0,1)
<i>RCU_I2Cx</i>	I2Cx clock (x = 0,1)
<i>RCU_WWDGT</i>	WWDGT clock
<i>RCU_BKP</i>	BKP clock
<i>RCU_PMU</i>	PMU clock
<i>RCU_DAC</i>	DAC clock
<i>RCU_RTC</i>	RTC clock
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 3-539. 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 3-528. Enum rcu_periph_enum
<i>RCU_GPIOx</i>	GPIOx ports clock (x = A,B,C,D,E,F)
<i>RCU_DMAx</i>	DMAx clock (x = 0, 1)
<i>RCU_CRC</i>	CRC clock
<i>RCU_SYSCFG</i>	SYSCFG clock
<i>RCU_CMP</i>	CMP clock
<i>RCU_ADCx</i>	ADCx clock (x = 0, 1)
<i>RCU_TIMERx</i>	TIMERx clock (x = 0,1,5,6,7,19,20)
<i>RCU_SPIx</i>	SPIx clock (x = 0,1)
<i>RCU_USARTx</i>	USARTx clock (x = 0,1,2)
<i>RCU_MFCOM</i>	MFCOM clock
<i>RCU_TRIGSEL</i>	TRIGSEL clock
<i>RCU_CANx</i>	CANx clock (x = 0,1)
<i>RCU_I2Cx</i>	I2Cx clock (x = 0,1)
<i>RCU_WWDGT</i>	WWDGT clock
<i>RCU_BKP</i>	BKP clock
<i>RCU_PMU</i>	PMU clock
<i>RCU_DAC</i>	DAC clock
<i>RCU_RTC</i>	RTC clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the USART0 clock */
```

```
rcu_periph_clock_disable(RCU_USART0);
```

rcu_periph_reset_enable

The description of rcu_periph_reset_enable is shown as below:

Table 3-540. 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 3-530. Enum rcu_periph_reset_enum
<i>RCU_GPIOxRST</i>	reset GPIO ports clock (x = A,B,C,D,E,F)
<i>RCU_DMAxRST</i>	reset DMAx clock (x = 0,1)
<i>RCU_DMAMUXRST</i>	reset DMAMUX clock
<i>RCU_MFCOMRST</i>	reset MFCOM clock
<i>RCU_CRCRST</i>	reset CRC clock
<i>RCU_SYSCFGRST</i>	reset SYSCFG clock
<i>RCU_CMPRST</i>	reset CMP clock
<i>RCU_ADCxRST</i>	reset ADCx clock (x = 0,1)
<i>RCU_TIMERxRST</i>	reset TIMERx clock (x = 0,1,5,6,7,19,20)
<i>RCU_SPIxRST</i>	reset SPIx clock (x = 0,1)
<i>RCU_USARTxRST</i>	reset USARTx clock (x = 0,1,2)
<i>RCU_CANxRST</i>	reset CANx clock (x = 0,1)
<i>RCU_I2CxRST</i>	reset I2Cx clock (x = 0,1)
<i>RCU_WWDGTRST</i>	reset WWDGT clock
<i>RCU_PMURST</i>	reset PMU clock
<i>RCU_DACRST</i>	reset DAC clock
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 3-541. 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 3-530. Enum rcu_periph_reset_enum

<i>RCU_GPIOxRST</i>	reset GPIO ports clock (x = A,B,C,D,E,F)
<i>RCU_DMAxRST</i>	reset DMAx clock (x = 0,1)
<i>RCU_DMAMUXRST</i>	reset DMAMUX clock
<i>RCU_MFCOMRST</i>	reset MFCOM clock
<i>RCU_CRCRST</i>	reset CRC clock
<i>RCU_SYSCFGRST</i>	reset SYSCFG clock
<i>RCU_CMPRST</i>	reset CMP clock
<i>RCU_ADCxRST</i>	reset ADCx clock (x = 0,1)
<i>RCU_TIMERxRST</i>	reset TIMERx clock (x = 0,1,5,6,7,19,20)
<i>RCU_SPIxRST</i>	reset SPIx clock (x = 0,1)
<i>RCU_USARTxRST</i>	reset USARTx clock (x = 0,1,2)
<i>RCU_CANxRST</i>	reset CANx clock (x = 0,1)
<i>RCU_I2CxRST</i>	reset I2Cx clock (x = 0,1)
<i>RCU_WWDGTRST</i>	reset WWDGT clock
<i>RCU_PMURST</i>	reset PMU clock
<i>RCU_DACRST</i>	reset DAC clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 reset */
```

```
rcu_periph_reset_disable(RCU_SPI0RST);
```

rcu_periph_clock_sleep_enable

The description of rcu_periph_clock_sleep_enable is shown as below:

Table 3-542. 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 3-529. Enum rcu_periph_sleep_enum
<i>RCU_FMC_SLP</i>	FMC clock
<i>RCU_SRAM_SLP</i>	SRAM clock
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 3-543. 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 3-529. Enum rcu_periph_sleep_enum
RCU_SRAM_SLP	SRAM clock
RCU_FMC_SLP	FMC clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

rcu_bkp_reset_enable

The description of rcu_bkp_reset_enable is shown as below:

Table 3-544. 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 3-545. Function rcu_bkp_reset_disable

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

Example:

```
/* disable the BKP domain reset */

rcu_bkp_reset_disable();
```

rcu_system_clock_source_config

The description of rcu_system_clock_source_config is shown as below:

Table 3-546. 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_IRC8M</i>	select CK_IRC8M as the CK_SYS source
<i>RCU_CKSYSSRC_HXTAL</i>	select CK_HXTAL as the CK_SYS source
<i>RCU_CKSYSSRC_PLL</i>	select CK_PLL as the CK_SYS source
Output parameter{out}	

-	-
Return value	
-	-

Example:

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

rcu_system_clock_source_get

The description of rcu_system_clock_source_get is shown as below:

Table 3-547. Function rcu_system_clock_source_get

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

Example:

```
uint32_t temp_cksys_status;

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

rcu_ahb_clock_config

The description of rcu_ahb_clock_config is shown as below:

Table 3-548. 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
<i>RCU_AHB_CKSYS_DI</i>	select CK_SYS / x, (x = 1, 2, 4, 8, 16, 64, 128, 256, 512)

Vx	
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 3-549. 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>IVx</i>	select (CK_AHB / x) as CK_APB1 (x = 1,2,4,8,16)
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 3-550. 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_DIVx</i>	select (CK_AHB / x) as CK_APB2 clock (x = 1,2,4,8,16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB2 */
```

```
rcu_apb2_clock_config(RCU_APB2_CKAHB_DIV8);
```

rcu_ckout_config

The description of rcu_ckout_config is shown as below:

Table 3-551. Function rcu_ckout_config

Function name	rcu_ckout_config
Function prototype	void rcu_ckout_config(uint32_t ckout_src, uint32_t ckout_div);
Function descriptions	configure the CK_OUT clock source and division factor
Precondition	-
The called functions	-
Input parameter{in}	
ckout_src	CK_OUT clock source selection
<i>RCU_CKOUTSRC_NONE</i>	no clock selected
<i>RCU_CKOUTSRC_IRC40K</i>	select high speed 40K internal oscillator clock
<i>RCU_CKOUTSRC_LXTAL</i>	select LXTAL clock
<i>RCU_CKOUTSRC_CKSYS</i>	select system clock CK_SYS
<i>RCU_CKOUTSRC_IRC8M</i>	select high speed 8M internal oscillator clock
<i>RCU_CKOUTSRC_HXTAL</i>	select HXTAL clock
<i>RCU_CKOUTSRC_CKPLL_DIV1</i>	select CK_PLL clock
<i>RCU_CKOUTSRC_CKPLL_DIV2</i>	select (CK_PLL / 2) clock
Input parameter{in}	
ckout_div	CK_OUT divider

<i>RCU_CKOUT_DIVx</i>	CK_OUT is divided by x(x = 1,2,4,8,16,32,64,128)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the HXTAL as CK_OUT clock source */
```

```
rcu_ckout_config(RCU_CKOUTSRC_HXTAL, RCU_CKOUT_DIV1);
```

rcu_pll_config

The description of rcu_pll_config is shown as below:

Table 3-552. Function rcu_pll_config

Function name	rcu_pll_config
Function prototype	void rcu_pll_config(uint32_t pll_src, uint32_t pll_mul);
Function descriptions	configure the main PLL clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_src	PLL clock source selection
<i>RCU_PLLSRC_IRC8M_DIV2</i>	IRC8M/2 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_mul	PLL clock multiplication factor
<i>RCU_PLL_MULx</i>	PLL source clock * x (x = 2..32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PLL */
```

```
rcu_pll_config(RCU_PLLSRC_HXTAL, RCU_PLL_MUL10);
```

rcu_double_pll_enable

The description of rcu_double_pll_enable is shown as below:

Table 3-553. Function rcu_double_pll_enable

Function name	rcu_double_pll_enable
----------------------	-----------------------

Function prototype	void rcu_double_pll_enable(void);
Function descriptions	enable double PLL clock
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable double PLL clock */
rcu_double_pll_enable();
```

rcu_double_pll_disable

The description of rcu_double_pll_disable is shown as below:

Table 3-554. Function rcu_double_pll_disable

Function name	rcu_double_pll_disable
Function prototype	void rcu_double_pll_disable(void);
Function descriptions	disable double PLL clock
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable double PLL clock */
rcu_double_pll_disable();
```

rcu_system_reset_enable

The description of rcu_system_reset_enable is shown as below:

Table 3-555. Function rcu_system_reset_enable

Function name	rcu_system_reset_enable
Function prototype	void rcu_system_reset_enable(uint32_t reset_source);

Function descriptions	enable RCU system reset
Precondition	-
The called functions	-
Input parameter{in}	
reset_source	reset source
<i>RCU_SYSRST_LOCKUP</i>	CPU Lock-Up reset
<i>RCU_SYSRST_LVD</i>	low voltage detection reset
<i>RCU_SYSRST_ECC</i>	ECC 2 bits error reset
<i>RCU_SYSRST_LOH</i>	lost of HXTAL reset
<i>RCU_SYSRST_LOP</i>	lost of PLL reset
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RCU CPU Lock-Up reset */
```

```
rcu_system_reset_enable(RCU_SYSRST_LOCKUP);
```

rcu_system_reset_disable

The description of rcu_system_reset_disable is shown as below:

Table 3-556. Function rcu_system_reset_disable

Function name	rcu_system_reset_disable
Function prototype	void rcu_system_reset_disable(uint32_t reset_source);
Function descriptions	disable RCU system reset
Precondition	-
The called functions	-
Input parameter{in}	
reset_source	reset source
<i>RCU_SYSRST_LOCKUP</i>	CPU Lock-Up reset
<i>RCU_SYSRST_LVD</i>	low voltage detection reset
<i>RCU_SYSRST_ECC</i>	ECC 2 bits error reset
<i>RCU_SYSRST_LOH</i>	lost of HXTAL reset
<i>RCU_SYSRST_LOP</i>	lost of PLL reset
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RCU CPU Lock-Up reset */
```

```
rcu_system_reset_disable(RCU_SYSRST_LOCKUP);
```

rcu_adc_clock_config

The description of rcu_adc_clock_config is shown as below:

Table 3-557. Function rcu_adc_clock_config

Function name	rcu_adc_clock_config
Function prototype	void rcu_adc_clock_config(uint32_t adc_psc);
Function descriptions	configure the ADC clock prescaler selection
Precondition	-
The called functions	-
Input parameter{in}	
adc_psc	ADC clock prescaler selection
<i>RCU_CKADC_CK_AHB</i> <i>_DIVx</i>	ADC prescaler select CK_AHB / (x) (x = 2,3,...,32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the ADC prescaler factor */
```

```
rcu_adc_clock_config(RCU_CKADC_CK_AHB_DIV2);
```

rcu_rtc_clock_config

The description of rcu_rtc_clock_config is shown as below:

Table 3-558. 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_RTCSRC_NONE</i>	no clock selected
<i>RCU_RTCSRC_LXTAL</i>	CK_LXTAL selected as RTC source clock
<i>RCU_RTCSRC_IRC40</i> <i>K</i>	CK_IRC40K selected as RTC source clock
<i>RCU_RTCSRC_HXTAL</i>	CK_HXTAL/128 selected as RTC source clock

<code>_DIV_128</code>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

rcu_usart_clock_config

The description of `rcu_usart_clock_config` is shown as below:

Table 3-559. Function `rcu_usart_clock_config`

Function name	<code>rcu_usart_clock_config</code>
Function prototype	<code>void rcu_usart_clock_config(uint32_t usart_periph, uint32_t usart_clock_source);</code>
Function descriptions	configure the USART clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USART peripheral
<code>USARTx</code>	<code>USARTx</code> ($x = 0, 1, 2$)
usart_clock_source	USART clock source selection
<code>RCU_USARTSRC_HXTAL</code>	HXTAL clock selected as USART source clock
<code>RCU_USARTSRC_CKSYS</code>	system clock selected as USART source clock
<code>RCU_USARTSRC_LXTAL</code>	LXTAL clock selected as USART source clock
<code>RCU_USARTSRC_IRC8M</code>	IRC8M clock selected as USART source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the LXTAL as USART0 clock */
rcu_usart_clock_config(USART0, RCU_USARTSRC_LXTAL);
```


rcu_can_clock_config

The description of rcu_can_clock_config is shown as below:

Table 3-560. Function rcu_can_clock_config

Function name	rcu_can_clock_config
Function prototype	void rcu_can_clock_config(uint32_t can_periph, uint32_t can_clock_source);
Function descriptions	configure the CAN clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
<i>CANx</i>	CANx(x = 0,1)
Input parameter{in}	
can_clock_source	CAN clock source
<i>RCU_CANSRC_HXTAL</i>	HXTAL clock selected as CAN source clock
<i>RCU_CANSRC_PCLK2</i>	PCLK2 clock selected as CAN source clock
<i>RCU_CANSRC_PCLK2_DIV_2</i>	PCLK2/2 clock selected as CAN source clock
<i>RCU_CANSRC_IRC8M</i>	IRC8M clock selected as CAN source clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CAN clock source selection */
```

```
rcu_can_clock_config(CAN0, RCU_CANSRC_IRC8M);
```

rcu_lxtal_drive_capability_config

The description of rcu_lxtal_drive_capability_config is shown as below:

Table 3-561. 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_LXTAL_LOWDRI</i>	lower driving capability

<i>RCU_LXTAL_MED_LO</i> <i>WDRI</i>	medium low driving capability
<i>RCU_LXTAL_MED_HI</i> <i>GHDRI</i>	medium high driving capability
<i>RCU_LXTAL_HIGHDRI</i>	higher driving capability
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the LXTAL lower driving capability */
```

```
rcu_lxtal_drive_capability_config(RCU_LXTAL_LOWDRI);
```

rcu_osci_stab_wait

The description of rcu_osci_stab_wait is shown as below:

Table 3-562. 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 3-535. Enum rcu_osci_type_enum
<i>RCU_HXTAL</i>	high speed crystal oscillator(HXTAL)
<i>RCU_LXTAL</i>	low speed crystal oscillator(LXTAL)
<i>RCU_IRC8M</i>	internal 8M RC oscillators(IRC8M)
<i>RCU_IRC40K</i>	internal 40K RC oscillator(IRC40K)
<i>RCU_PLL_CK</i>	phase locked loop(PLL)
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 3-563. 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 3-535. Enum rcu_osci_type_enum
<i>RCU_HXTAL</i>	high speed crystal oscillator(HXTAL)
<i>RCU_LXTAL</i>	low speed crystal oscillator(LXTAL)
<i>RCU_IRC8M</i>	internal 8M RC oscillators(IRC8M)
<i>RCU_IRC40K</i>	internal 40K RC oscillator(IRC40K)
<i>RCU_PLL_CK</i>	phase locked loop(PLL)
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 3-564. 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 3-535. Enum rcu_osci_type_enum
<i>RCU_HXTAL</i>	high speed crystal oscillator(HXTAL)
<i>RCU_LXTAL</i>	low speed crystal oscillator(LXTAL)
<i>RCU_IRC8M</i>	internal 8M RC oscillators(IRC8M)
<i>RCU_IRC40K</i>	internal 40K RC oscillator(IRC40K)
<i>RCU_PLL_CK</i>	phase locked loop(PLL)
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 3-565. 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 3-535. 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 3-566. 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 3-535. 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_frequency_scale_select

The description of rcu_hxtal_frequency_scale_select is shown as below:

Table 3-567. Function

Function name	rcu_hxtal_frequency_scale_select
Function prototype	void rcu_hxtal_frequency_scale_select(uint32_t hxtal_scal);
Function descriptions	HXTAL frequency scale select
Precondition	-
The called functions	-
Input parameter{in}	
hxtal_scal	HXTAL frequency scale
<i>HXTAL_SCALE_2M_TO_8M</i>	HXTAL scale is 2-8MHz
<i>HXTAL_SCALE_8M_TO_40M</i>	HXTAL scale is 8-40MHz
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* HXTAL frequency scale select */
```

```
rcu_hxtal_frequency_scale_select(HXTAL_SCALE_2M_TO_8M);
```

rcu_hxtal_prediv_config

The description of rcu_hxtal_prediv_config is shown as below:

Table 3-568. Function rcu_hxtal_prediv_config

Function name	rcu_hxtal_prediv_config
Function prototype	void rcu_hxtal_prediv_config(uint32_t hxtal_prediv);

Function descriptions	configure the HXTAL divider used as input of PLL
Precondition	-
The called functions	-
Input parameter{in}	
hxtal_prediv	HXTAL divider previous PLL
RCU_PREDV_DIVx	HXTAL divided x used as input of PLL (x = 1..16)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the HXTAL divider used as input of PLL */
```

```
rcu_hxtal_prediv_config(RCU_PREDV_DIV1);
```

rcu_irc8m_adjust_value_set

The description of rcu_irc8m_adjust_value_set is shown as below:

Table 3-569. Function rcu_irc8m_adjust_value_set

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

Example:

```
/* set the IRC8M adjust value */
```

```
rcu_irc8m_adjust_value_set(0x10);
```

rcu_hxtal_clock_monitor_enable

The description of rcu_hxtal_clock_monitor_enable is shown as below:

Table 3-570. 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 3-571. 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_lxtal_clock_monitor_enable

The description of rcu_lxtal_clock_monitor_enable is shown as below:

Table 3-572. Function rcu_lxtal_clock_monitor_enable

Function name	rcu_lxtal_clock_monitor_enable
Function prototype	void rcu_lxtal_clock_monitor_enable(void);
Function descriptions	enable the LXTAL clock monitor

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

Example:

```
/* enable the LXTAL clock monitor */
```

```
rcu_lxtal_clock_monitor_enable();
```

rcu_lxtal_clock_monitor_disable

The description of rcu_lxtal_clock_monitor_disable is shown as below:

Table 3-573. Function rcu_lxtal_clock_monitor_disable

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

Example:

```
/* disable the LXTAL clock monitor */
```

```
rcu_lxtal_clock_monitor_disable();
```

rcu_voltage_key_unlock

The description of rcu_voltage_key_unlock is shown as below:

Table 3-574. 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 3-575. Function rcu_deepsleep_voltage_set

Function name	rcu_deepsleep_voltage_set
Function prototype	void rcu_deepsleep_voltage_set(uint32_t dsvol);
Function descriptions	deep-sleep mode voltage select
Precondition	-
The called functions	-
Input parameter{in}	
dsvol	deep sleep mode voltage
RCU_DEEPSLEEP_V_0_8	the core voltage is 0.8V
RCU_DEEPSLEEP_V_0_9	the core voltage is 0.9V
RCU_DEEPSLEEP_V_1_0	the core voltage is 1.0V
RCU_DEEPSLEEP_V_1_1	the core voltage is 1.1V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select 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 3-576. 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 and peripheral clock frequency
Precondition	-
The called functions	-
Input parameter{in}	
clock	the clock frequency which to get, refer to Table 3-536. Enum rcu_clock_freq_enum
CK_SYS	system clock frequency
CK_AHB	AHB clock frequency
CK_APB1	APB1 clock frequency
CK_APB2	APB2 clock frequency
CK_USART0	USART0 clock frequency
CK_USART1	USART1 clock frequency
CK_USART2	USART2 clock frequency
Output parameter{out}	
-	
Return value	
uint32_t	clock frequency of system, AHB, APB1, APB2, ADC or USRAT

Example:

```
uint32_t temp_freq;

/* get the system clock frequency */

temp_freq = rcu_clock_freq_get(CK_SYS);
```

rcu_flag_get

The description of rcu_flag_get is shown as below:

Table 3-577. 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 3-531. Enum rcu_flag_enum

<i>RCU_FLAG_IRC8MST</i> <i>B</i>	IRC8M stabilization flag
<i>RCU_FLAG_HXTALST</i> <i>B</i>	HXTAL stabilization flag
<i>RCU_FLAG_PLLSTB</i>	PLL stabilization flag
<i>RCU_FLAG_LXTALST</i> <i>B</i>	LXTAL stabilization flag
<i>RCU_FLAG_IRC40KST</i> <i>B</i>	IRC40K stabilization flag
<i>RCU_FLAG_BORRST</i>	BOR reset flag
<i>RCU_FLAG_LOCKUP</i> <i>RST</i>	CPU LOCK UP error reset flag
<i>RCU_FLAG_LVDRST</i>	low voltage detect error reset flag
<i>RCU_FLAG_ECCRST</i>	2 bits ECC error reset flag
<i>RCU_FLAG_LOHRST</i>	lost of HXTAL error reset flag
<i>RCU_FLAG_LOPRST</i>	lost of PLL error reset flag
<i>RCU_FLAG_V11RST</i>	1.1V domain Power reset flag
<i>RCU_FLAG_OBLRST</i>	option byte loader reset flag
<i>RCU_FLAG_EPRST</i>	external PIN reset flag
<i>RCU_FLAG_PORRST</i>	power reset flag
<i>RCU_FLAG_SWRST</i>	software reset flag
<i>RCU_FLAG_FWDGTR</i> <i>ST</i>	FWDGT reset flag
<i>RCU_FLAG_WWDGTR</i> <i>ST</i>	WWDGT reset flag
<i>RCU_FLAG_LPRST</i>	low-power reset flag
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 3-578. 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 3-579. 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 3-532. Enum rcu_int_flag_enum
<i>RCU_INT_FLAG_IRC40KSTB</i>	IRC40K stabilization interrupt flag
<i>RCU_INT_FLAG_LXTALSTB</i>	LXTAL stabilization interrupt flag
<i>RCU_INT_FLAG_IRC8MSTB</i>	IRC8M stabilization interrupt flag
<i>RCU_INT_FLAG_HXTALALSTB</i>	HXTAL stabilization interrupt flag
<i>RCU_INT_FLAG_PLLSBTB</i>	PLL stabilization interrupt flag
<i>RCU_INT_FLAG_LCKM</i>	LXTAL stabilization interrupt flag
<i>RCU_INT_FLAG_PLLM</i>	PLL clock monitor interrupt flag
<i>RCU_INT_FLAG_CKM</i>	HXTAL clock stuck interrupt flag
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 3-580. 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 3-533. Enum rcu_int_flag_clear_enum
RCU_INT_FLAG_IRC40KSTB_CLR	IRC40K stabilization interrupt flag clear
RCU_INT_FLAG_LXTALSTB_CLR	LXTAL stabilization interrupt flag clear
RCU_INT_FLAG_IRC8MSTB_CLR	IRC8M stabilization interrupt flag clear
RCU_INT_FLAG_HXTALSTB_CLR	HXTAL stabilization interrupt flag clear
RCU_INT_FLAG_PLLSTB_CLR	PLL stabilization interrupt flag clear
RCU_INT_FLAG_LCKM_CLR	LXTAL stabilization interrupt flag clear
RCU_INT_FLAG_PLLM_CLR	PLL clock monitor interrupt flag clear
RCU_INT_FLAG_CKM_CLR	HXTAL clock stuck interrupt flag clear
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the interrupt HXTAL stabilization interrupt flag */
```

```
rcu_interrupt_flag_clear(RCU_INT_FLAG_HXTALSTB_CLR);
```

rcu_interrupt_enable

The description of rcu_interrupt_enable is shown as below:

Table 3-581. Function rcu_interrupt_enable

Function name	rcu_interrupt_enable
Function prototype	void rcu_interrupt_enable(rcu_int_enum interrupt);
Function descriptions	enable the stabilization interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	clock stabilization interrupt, refer to Table 3-534. Enum rcu_int_enum
<i>RCU_INT_IRC40KSTB</i>	IRC40K stabilization interrupt enable
<i>RCU_INT_LXTALSTB</i>	LXTAL stabilization interrupt enable
<i>RCU_INT_IRC8MSTB</i>	IRC8M stabilization interrupt enable
<i>RCU_INT_HXTALSTB</i>	HXTAL stabilization interrupt enable
<i>RCU_INT_PLLSTB</i>	PLL stabilization interrupt enable
<i>RCU_INT_LCKM</i>	LXTAL clock monitor interrupt
<i>RCU_INT_PLLM</i>	PLL clock monitor interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the HXTAL stabilization interrupt */
```

```
rcu_interrupt_enable(RCU_INT_HXTALSTB);
```

rcu_interrupt_disable

The description of rcu_interrupt_disable is shown as below:

Table 3-582. 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 3-534. Enum rcu_int_enum
<i>RCU_INT_IRC40KSTB</i>	IRC40K stabilization interrupt enable
<i>RCU_INT_LXTALSTB</i>	LXTAL stabilization interrupt enable

<i>RCU_INT_IRC8MSTB</i>	IRC8M stabilization interrupt enable
<i>RCU_INT_HXTALSTB</i>	HXTAL stabilization interrupt enable
<i>RCU_INT_PLLSTB</i>	PLL stabilization interrupt enable
<i>RCU_INT_LCKM</i>	LXTAL clock monitor interrupt
<i>RCU_INT_PLLM</i>	PLL clock monitor interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

3.19. RTC

The Real-time Clock (RTC) is usually used as a clock-calendar. The ones in the Backup Domain consist of a 32-bit up-counter, an alarm, a prescaler, a divider and the RTC clock configuration register. The RTC registers are listed in chapter [3.19.1](#), the FWDGT firmware functions are introduced in chapter [3.19.2](#).

3.19.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

Table 3-583. RTC Registers

Registers	Descriptions
RTC_INTEN	interrupt enable register
RTC_CTL	control register
RTC_PSCH	prescaler high register
RTC_PSCL	prescaler low register
RTC_DIVH	divider high register
RTC_DIVL	divider low register
RTC_CNTH	counter high register
RTC_CNTL	counter low register
RTC_ALRMH	alarm high register
RTC_ALRML	alarm low register

3.19.2. Descriptions of Peripheral functions

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

Table 3-584. RTC firmware function

Function name	Function description
rtc_configuration_mode_enter	enter RTC configuration mode
rtc_configuration_mode_exit	exit RTC configuration mode
rtc_lwoff_wait	wait RTC last write operation finished flag set
rtc_register_sync_wait	wait RTC registers synchronized flag set
rtc_counter_get	get RTC counter value
rtc_counter_set	set RTC counter value
rtc_prescaler_set	set RTC prescaler value
rtc_alarm_config	set RTC alarm value
rtc_divider_get	get RTC divider value
rtc_interrupt_enable	enable RTC interrupt
rtc_interrupt_disable	disable RTC interrupt
rtc_flag_get	get RTC flag status
rtc_flag_clear	clear RTC flag status
rtc_interrupt_flag_get	get RTC interrupt flag status
rtc_interrupt_flag_clear	clear RTC interrupt flag status

rtc_configuration_mode_enter

The description of rtc_configuration_mode_enter is shown as below:

Table 3-585. Function rtc_configuration_mode_enter

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

Example:

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

rtc_configuration_mode_exit

The description of rtc_configuration_mode_exit is shown as below:

Table 3-586. Function rtc_configuration_mode_exit

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

Example:

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

rtc_lwoff_wait

The description of rtc_lwoff_wait is shown as below:

Table 3-587. Function rtc_lwoff_wait

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

Example:

```
/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait();

/* enable the RTC second interrupt */
rtc_interrupt_enable(RTC_INT_SECOND);
```

rtc_register_sync_wait

The description of rtc_register_sync_wait is shown as below:

Table 3-588. Function rtc_register_sync_wait

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

Example:

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

rtc_counter_get

The description of rtc_counter_get is shown as below:

Table 3-589. Function rtc_counter_get

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

Example:

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

rtc_counter_set

The description of rtc_counter_set is shown as below:

Table 3-590. Function rtc_counter_set

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

Example:

```

/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait();

/* set counter value to 0xFFFF */
rtc_counter_set(0xFFFF);

```

rtc_prescaler_set

The description of rtc_prescaler_set is shown as below:

Table 3-591. Function rtc_prescaler_set

Function name	rtc_interrupt_rtc_prescaler_set
Function prototype	void rtc_prescaler_set(uint32_t psc);
Function descriptions	set RTC prescaler value
Precondition	before using this function, you must call rtc_lwoff_wait() function (wait until LWOFF flag is set)
The called functions	rtc_configuration_mode_enter / rtc_configuration_mode_exit
Input parameter{in}	
psc	RTC prescaler value (0-0x000F FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* wait until last write operation on RTC registers has finished */

```

```
rtc_lwoff_wait( );

/* set RTC prescaler value to 0x7FFFF */

rtc_prescaler_set(0x7FFFF);
```

rtc_alarm_config

The description of rtc_alarm_config is shown as below:

Table 3-592. Function rtc_alarm_config

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

Example:

```
/* wait until last write operation on RTC registers has finished */

rtc_lwoff_wait();

/* set alarm value to 0xFFFF */

rtc_alarm_config(0xFFFF);
```

rtc_divider_get

The description of rtc_divider_get is shown as below:

Table 3-593. Function rtc_divider_get

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

Return value	
uint32_t	the value of RTC divider

Example:

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

```
uint32_t rtc_divider_value;
```

```
rtc_divider_value = rtc_divider_get();
```

rtc_interrupt_enable

The description of rtc_interrupt_enable is shown as below:

Table 3-594. Function rtc_interrupt_enable

Function name	rtc_interrupt_enable
Function prototype	void rtc_interrupt_enable(uint32_t interrupt);
Function descriptions	enable RTC interrupt
Precondition	before using this function, you must call rtc_lwoff_wait () function (wait until LWOFF flag is set).
The called functions	-
Input parameter{in}	
interrupt	specify which RTC interrupt to enable
<i>RTC_INT_SECOND</i>	second interrupt
<i>RTC_INT_ALARM</i>	alarm interrupt
<i>RTC_INT_OVERFLOW</i>	overflow interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* wait until last write operation on RTC registers has finished */
```

```
rtc_lwoff_wait();
```

```
/* enable the RTC second interrupt */
```

```
rtc_interrupt_enable(RTC_INT_SECOND);
```

rtc_interrupt_disable

The description of rtc_interrupt_disable is shown as below:

Table 3-595. Function rtc_interrupt_disable

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

Function descriptions	disable RTC interrupt
Precondition	before using this function, you must call rtc_lwoff_wait () function (wait until LWOFF flag is set).
The called functions	-
Input parameter{in}	
interrupt	specify which RTC interrupt to disable
<i>RTC_INT_SECOND</i>	second interrupt
<i>RTC_INT_ALARM</i>	alarm interrupt
<i>RTC_INT_OVERFLOW</i>	overflow interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* wait until last write operation on RTC registers has finished */
rtc_lwoff_wait();

/* disable the RTC second interrupt */
rtc_interrupt_disable(RTC_INT_SECOND);

```

rtc_flag_get

The description of rtc_flag_get is shown as below:

Table 3-596. Function rtc_flag_get

Function name	rtc_flag_get
Function prototype	FlagStatus rtc_flag_get(uint32_t flag);
Function descriptions	get RTC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC flag status to get
<i>RTC_FLAG_SECOND</i>	second interrupt flag
<i>RTC_FLAG_ALARM</i>	alarm interrupt flag
<i>RTC_FLAG_OVERFLOW</i>	overflow interrupt flag
<i>RTC_FLAG_RSYN</i>	registers synchronized flag
<i>RTC_FLAG_LWOF</i>	last write operation finished flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the RTC overflow interrupt status */

FlagStatus alarm_status;

alarm_status = rtc_flag_get (RTC_FLAG_ALARM);
```

rtc_flag_clear

The description of rtc_flag_clear is shown as below:

Table 3-597. Function rtc_flag_clear

Function name	rtc_flag_clear
Function prototype	void rtc_flag_clear(uint32_t flag);
Function descriptions	clear RTC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC flag status to clear
<i>RTC_FLAG_SECOND</i>	second interrupt flag
<i>RTC_FLAG_ALARM</i>	alarm interrupt flag
<i>RTC_FLAG_OVERFLOW</i>	overflow interrupt flag
<i>RTC_FLAG_RSYN</i>	registers synchronized flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the RTC alarm flag */

rtc_flag_clear(RTC_FLAG_ALARM);
```

rtc_interrupt_flag_get

The description of rtc_interrupt_flag_get is shown as below:

Table 3-598. Function rtc_interrupt_flag_get

Function name	rtc_interrupt_flag_get
Function prototype	FlagStatus rtc_interrupt_flag_get(uint32_t flag);
Function descriptions	get RTC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC interrupt flag status to get

<i>RTC_INT_FLAG_SEC</i> <i>OND</i>	second interrupt flag
<i>RTC_INT_FLAG_ALAR</i> <i>M</i>	alarm interrupt flag
<i>RTC_INT_FLAG_OVE</i> <i>RFLOW</i>	overflow interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the RTC overflow interrupt status */
```

```
FlagStatus alarm_status;
```

```
alarm_status = rtc_interrupt_flag_get(RTC_INT_FLAG_ALARM);
```

rtc_interrupt_flag_clear

The description of `rtc_interrupt_flag_clear` is shown as below:

Table 3-599. Function `rtc_interrupt_flag_clear`

Function name	<code>rtc_interrupt_flag_clear</code>
Function prototype	<code>void rtc_interrupt_flag_clear(uint32_t flag);</code>
Function descriptions	clear RTC interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which RTC interrupt flag status to clear
<i>RTC_INT_FLAG_SEC</i> <i>OND</i>	second interrupt flag
<i>RTC_INT_FLAG_ALAR</i> <i>M</i>	alarm interrupt flag
<i>RTC_INT_FLAG_OVE</i> <i>RFLOW</i>	overflow interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the RTC alarm interrupt flag */
```

```
rtc_interrupt_flag_clear(RTC_FLAG_ALARM);
```


3.20. 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 [3.20.1](#), the SPI/I2S firmware functions are introduced in chapter [3.20.2](#).

3.20.1. Descriptions of Peripheral registers

SPI/I2S registers are listed in the table shown as below:

Table 3-600. 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

3.20.2. Descriptions of Peripheral functions

SPI/I2S firmware functions are listed in the table shown as below:

Table 3-601. SPI/I2S firmware function

Function name	Function description
spi_i2s_deinit	reset SPI and I2S peripheral
spi_struct_para_init	initialize the parameters of SPI struct with the default values
spi_init	initialize SPI peripheral parameter
spi_enable	enable SPI
spi_disable	disable SPI
i2s_init	initialize I2S peripheral parameter
i2s_psc_config	configure I2S peripheral prescaler
i2s_enable	enable I2S
i2s_disable	disable I2S
spi_nss_output_enable	enable SPI NSS output
spi_nss_output_disable	disable SPI NSS output
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

Function name	Function description
spi_dma_disable	disable SPI DMA function
spi_i2s_data_frame_format_config	configure SPI/I2S data frame format
spi_i2s_data_transmit	SPI transmit data
spi_i2s_data_receive	SPI receive data
spi_bidirectional_transfer_config	configure SPI bidirectional transfer direction
spi_i2s_format_error_clear	clear SPI/I2S 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_nssp_mode_enable	enable SPI NSS pulse mode
spi_nssp_mode_disable	disable SPI NSS pulse mode
spi_quad_enable	enable quad wire SPI
spi_quad_disable	disable quad wire SPI
spi_quad_write_enable	enable quad wire SPI write
spi_quad_read_enable	enable quad wire SPI read
spi_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_interrupt_enable	enable SPI and I2S interrupt
spi_i2s_interrupt_disable	disable SPI and I2S interrupt
spi_i2s_interrupt_flag_get	get SPI and I2S interrupt status
spi_i2s_flag_get	get SPI and I2S flag status

Structure spi_parameter_struct

Table 3-602. 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 (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 3-603. 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
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 3-604. 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 structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
spi_struct	SPI parameter struct, the structure members can refer to members of the

	structure Table 3-602. spi_parameter_struct
Return value	
-	-

Example:

```

/* initialize the parameters of SPI struct */

spi_parameter_struct spi_struct;

spi_struct->device_mode = SPI_SLAVE;

spi_struct->trans_mode = SPI_TRANSMODE_FULLDUPLEX;

spi_struct->frame_size = SPI_FRAMESIZE_8BIT;

spi_struct->nss = SPI_NSS_HARD;

spi_struct->clock_polarity_phase = SPI_CK_PL_LOW_PH_1EDGE;

spi_struct->prescale = SPI_PSC_2;

spi_struct_para_init(&spi_struct);

```

spi_init

The description of spi_init is shown as below:

Table 3-605. 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 parameters
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1
Input parameter{in}	
spi_struct	SPI parameter initialization struct, the structure members can refer to members of the structure Table 3-602. 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_CK_PL_HIGH_PH_2EDGE;

spi_init_struct.nss = SPI_NSS_SOFT;

spi_init_struct.prescale = SPI_PSC_8;

spi_init_struct.endian = SPI_ENDIAN_MSB;

spi_init(SPI0, &spi_init_struct);

```

spi_enable

The description of spi_enable is shown as below:

Table 3-606. 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 3-607. Function spi_disable

Function name	spi_disable
Function prototype	void spi_disable(uint32_t spi_periph);
Function descriptions	disable SPIx
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 3-608. Function i2s_init

Function name	i2s_init
Function prototype	void i2s_init(uint32_t spi_periph, uint32_t mode, uint32_t standard, uint32_t ckpl);
Function descriptions	initialize I2S peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
<i>SPIx</i>	x=1
Input parameter{in}	
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> X	I2S master transmit mode
<i>I2S_MODE_MASTERRX</i> X	I2S master receive mode
Input parameter{in}	
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}	
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 3-609. Function i2s_psc_config

Function name	i2s_psc_config
Function prototype	void i2s_psc_config(uint32_t spi_periph, uint32_t audiosample, uint32_t frameformat, uint32_t mckout);
Function descriptions	configure I2S prescaler
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
spi_periph	I2S peripheral
<i>SPIx</i>	x=1
Input parameter{in}	
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_1</i> 92K	audio sample rate is 192KHz
Input parameter{in}	
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}	
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_enable

The description of i2s_enable is shown as below:

Table 3-610. 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>SPIx</i>	x=1
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* enable I2S1*/
i2s_enable(SPI1);
```

i2s_disable

The description of i2s_disable is shown as below:

Table 3-611. 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>SPIx</i>	x=1
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 3-612. 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
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 3-613. 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
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	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 3-614. Function spi_nss_internal_high

Function name	spi_nss_internal_high
Function prototype	void spi_nss_internal_high(uint32_t spi_periph);
Function descriptions	SPI NSS pin high level in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1

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 3-615. Function spi_nss_internal_low

Function name	spi_nss_internal_low
Function prototype	void spi_nss_internal_low(uint32_t spi_periph);
Function descriptions	SPI NSS pin low level in software mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1
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 3-616. 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 send or receive
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral

<i>SPIx</i>	<i>x</i> =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 3-617. 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>	<i>x</i> =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 3-618. 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 data frame format
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1
Input parameter{in}	
frame_format	SPI frame size
SPI_FRAME_SIZE_16BIT	SPI frame size is 16 bits
SPI_FRAME_SIZE_8BIT	SPI frame size is 8 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SPI1/I2S1 data frame format size is 16 bits */
```

```
spi_i2s_data_frame_format_config(SPI1, SPI_FRAME_SIZE_16BIT);
```

spi_i2s_data_transmit

The description of spi_i2s_data_transmit is shown as below:

Table 3-619. Function spi_i2s_data_transmit

Function name	spi_i2s_data_transmit
Function prototype	void spi_i2s_data_transmit(uint32_t spi_periph, uint16_t data);
Function descriptions	SPI transmit data
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx(x=0,1,2)	x=0,1
Input parameter{in}	
data	16-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 transmit data */
```

```
spi_i2s_data_transmit(SPI0, spi0_send_array[send_n]);
```

spi_i2s_data_receive

The description of spi_i2s_data_receive is shown as below:

Table 3-620. 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 */
```

```
spi0_receive_array[receive_n] = spi_i2s_data_receive(SPI0);
```

spi_bidirectional_transfer_config

The description of spi_bidirectional_transfer_config is shown as below:

Table 3-621. 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_</i>	SPI work in transmit-only mode

<i>TRANSMIT</i>	
<i>SPI_BIDIRECTIONAL_</i> <i>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);
```

spi_i2s_format_error_clear

The description of spi_i2s_format_error_clear is shown as below:

Table 3-622. 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 SPI/I2S format 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 SPI/I2S format error flag status */
```

```
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 3-623. Function spi_crc_polynomial_set

Function name	spi_crc_polynomial_set
Function prototype	void spi_crc_polynomial_set(uint32_t spi_periph, uint16_t crc_poly);
Function descriptions	set SPI CRC polynomial
Precondition	-

The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1
Input parameter{in}	
crc_poly	CRC polynomial value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SPI0 CRC polynomial */
uint16_t CRC_VALUE = 0x8;

spi_crc_polynomial_set(SPI0,CRC_VALUE);
```

spi_crc_polynomial_get

The description of spi_crc_polynomial_get is shown as below:

Table 3-624. Function spi_crc_polynomial_get

Function name	spi_crc_polynomial_get
Function prototype	uint16_t spi_crc_polynomial_get(uint32_t spi_periph);
Function descriptions	get SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit CRC polynomial (0-0xFFFF)

Example:

```
/* get SPI0 CRC polynomial */
uint16_t CRC_VALUE;

CRC_VALUE = spi_crc_polynomial_get(SPI0);
```

spi_crc_on

The description of spi_crc_on is shown as below:

Table 3-625. Function spi_crc_on

Function name	spi_crc_on
Function prototype	void spi_crc_on(uint32_t spi_periph);
Function descriptions	turn on CRC function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	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 3-626. Function spi_crc_off

Function name	spi_crc_off
Function prototype	void spi_crc_off(uint32_t spi_periph);
Function descriptions	turn off CRC function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	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 3-627. Function spi_crc_next

Function name	spi_crc_next
Function prototype	void spi_crc_next(uint32_t spi_periph);
Function descriptions	SPI next data is CRC value
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1
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 3-628. Function spi_crc_get

Function name	spi_crc_get
Function prototype	uint16_t spi_crc_get(uint32_t spi_periph, uint8_t crc);
Function descriptions	get SPI CRC send value or receive value
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1
Input parameter{in}	
crc	SPI crc value
SPI_CRC_TX	get transmit CRC value
SPI_CRC_RX	get receive CRC value
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit CRC value (0-0xFFFF)

Example:

```
/* get SPI0 CRC send value */
uint16_t value;
```

```
value = spi_crc_get(SPI0, SPI_CRC_TX);
```

spi_crc_error_clear

The description of spi_crc_error_clear is shown as below:

Table 3-629. 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 3-630. 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 3-631. 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_nssp_mode_enable

The description of spi_nssp_mode_enable is shown as below:

Table 3-632. Function spi_nssp_mode_enable

Function name	spi_ti_mode_enable
Function prototype	void spi_ti_mode_enable(uint32_t spi_periph);
Function descriptions	enable SPI NSS pulse 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 NSS pulse mode */
```

```
spi_nssp_mode_enable(SPI0);
```

spi_nssp_mode_disable

The description of spi_nssp_mode_disable is shown as below:

Table 3-633. Function spi_nssp_mode_disable

Function name	spi_nssp_mode_disable
Function prototype	void spi_nssp_mode_disable(uint32_t spi_periph);
Function descriptions	disable SPI NSS pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
<i>SPIx</i>	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 NSS pulse mode */
```

```
spi_nssp_mode_disable(SPI0);
```

spi_quad_enable

The description of spi_quad_enable is shown as below:

Table 3-634. Function spi_quad_enable

Function name	spi_quad_enable
Function prototype	void qspi_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 quad wire SPI */
```

```
spi_quad_enable (SPI0);
```

spi_quad_disable

The description of spi_quad_disable is shown as below:

Table 3-635. Function spi_quad_disable

Function name	spi_quad_disable
Function prototype	void 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 quad wire SPI */
```

```
spi_quad_disable (SPI0);
```

spi_quad_write_enable

The description of spi_quad_write_enable is shown as below:

Table 3-636. 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 quad wire SPI write */
```

spi_quad_write_enable (SPI0);

spi_quad_read_enable

The description of spi_quad_read_enable is shown as below:

Table 3-637. 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 quad wire SPI read */
```

```
spi_quad_read_enable (SPI0);
```

spi_quad_io23_output_enable

The description of spi_quad_io23_output_enable is shown as below:

Table 3-638. 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 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 3-639. 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 SPI_IO2 and SPI_IO3 pin output */
```

```
spi_quad_io23_output_disable(SPI0);
```

spi_i2s_interrupt_enable

The description of spi_i2s_interrupt_enable is shown as below:

Table 3-640. 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 3-641. 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 3-642. 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_CONFIGERR</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_FORMATERR</i>	format error interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SPI0 transmit buffer empty interrupt status */
```

```
FlagStatus Flag_interrupt = RESET;
```

```
Flag_interrupt = spi_i2s_interrupt_flag_get(SPI0, SPI_I2S_INT_FLAG_TBE);
```

spi_i2s_flag_get

The description of spi_i2s_flag_get is shown as below:

Table 3-643. 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>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
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SPI0 transmit buffer empty flag status */
```

```
FlagStatus Flag = RESET;
```

```
Flag = spi_i2s_flag_get(SPI0, SPI_FLAG_TBE);
```

3.21. SYSCFG

The SYSCFG registers are listed in chapter [3.21.1](#), and the SYSCFG firmware functions are introduced in chapter [3.21.2](#).

3.21.1. Descriptions of Peripheral registers

SYSCFG registers are listed in the table shown as below:

Table 3-644. SYSCFG Registers

Registers	Descriptions
SYSCFG_CFG0	system configuration register 0
SYSCFG_CFG1	system configuration register 1
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_CFG2	system configuration register 2
SYSCFG_STAT	system status register
SYSCFG_CFG3	system configuration register 3
SYSCFG_TIMERINSEL	TIMER input source selection register

3.21.2. Descriptions of Peripheral functions

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

Table 3-645. SYSCFG firmware function

Function name	Function description
syscfg_deinit	reset the SYSCFG registers
syscfg_exti_line_config	configure the GPIO pin as EXTI Line
syscfg_pin_remap_enable	enable remap pin function
syscfg_pin_remap_disable	disable remap pin function
syscfg_adc_ch_remap_config	configure ADC channel GPIO pin remap function
syscfg_timer_eti_sel	select TIMER external trigger source
syscfg_timer_bkin_select_trigsel	select TRIGSEL as TIMER break input source
syscfg_timer_bkin_select_gpio	select GPIO as TIMER break input source
syscfg_timer7_ch0n_select	select TIMER7 channel0 complementary input source
syscfg_lock_config	configure TIMER0/7/19/20 break input to the selected parameter connection
syscfg_flag_get	get SYSCFG flags
syscfg_flag_clear	clear SYSCFG flags
syscfg_interrupt_enable	enable SYSCFG interrupts
syscfg_interrupt_disable	disable SYSCFG interrupts
syscfg_interrupt_flag_get	get SYSCFG interrupt flag status
syscfg_sram_ecc_address_get	get the address where SRAM ECC error occur on
syscfg_sram_ecc_bit_get	get the bit which has SRAM ECC single error

syscfg_deinit

The description of syscfg_deinit is shown as below:

Table 3-646. 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 registers */
```

```
syscfg_deinit();
```

syscfg_exti_line_config

The description of syscfg_exti_line_config is shown as below:

Table 3-647. 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	x = A,B,C,D,E,F
Input parameter{in}	
exti_pin	specify the EXTI line
EXTI_SOURCE_PINx	for GPIOA\GPIOB\GPIOC\GPIOD\GPIOE, x = 0..15, for GPIOF, x = 0..7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PA0 pin as EXTI Line */
```

```
syscfg_exti_line_config(EXTI_SOURCE_GPIOA, EXTI_SOURCE_PIN0);
```

syscfg_pin_remap_enable

The description of syscfg_pin_remap_enable is shown as below:

Table 3-648. Function syscfg_pin_remap_enable

Function name	syscfg_pin_remap_enable
Function prototype	void syscfg_pin_remap_enable(uint32_t remap_pin);

Function descriptions	enable remap pin function
Precondition	-
The called functions	-
Input parameter{in}	
remap_pin	remap pin
<i>SYSCFG_PA9_PA12_REMAP</i>	PA9/PA12 pins are mapping on PA10/PA11 pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable PA9/PA12 remap to PA10/PA11 function */
syscfg_pin_remap_enable(SYSCFG_PA9_PA12_REMAP);
```

syscfg_pin_remap_disable

The description of syscfg_pin_remap_disable is shown as below:

Table 3-649. Function syscfg_pin_remap_disable

Function name	syscfg_pin_remap_disable
Function prototype	void syscfg_pin_remap_disable(void);
Function descriptions	disable remap pin function
Precondition	-
The called functions	-
Input parameter{in}	
remap_pin	remap pin
<i>SYSCFG_PA9_PA12_REMAP</i>	PA9/PA12 pins are mapping on PA10/PA11 pins
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable PA9/PA12 remap to PA10/PA11 function */
syscfg_pin_remap_disable(SYSCFG_PA9_PA12_REMAP);
```

syscfg_adc_ch_remap_config

The description of syscfg_adc_ch_remap_config is shown as below:

Table 3-650. Function syscfg_adc_ch_remap_config

Function name	syscfg_adc_ch_remap_config
Function prototype	void syscfg_adc_ch_remap_config(syscfg_adcx_chy_enum adcx_iny_remap, ControlStatus newvalue);
Function descriptions	configure ADC channel GPIO pin remap function
Precondition	-
The called functions	-
Input parameter{in}	
adcx_iny_remap	specify ADC channel
ADC1_IN14_REMAP	ADC1 channel 14 GPIO pin remap
ADC1_IN15_REMAP	ADC1 channel 15 GPIO pin remap
ADC0_IN8_REMAP	ADC0 channel 8 GPIO pin remap
ADC0_IN9_REMAP	ADC0 channel 9 GPIO pin remap
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC1 channel 14 GPIO pin remap function */
syscfg_adc_ch_remap_config (ADC1_IN14_REMAP, ENABLE);
```

syscfg_timer_eti_sel

The description of syscfg_timer_eti_sel is shown as below:

Table 3-651. Function syscfg_timer_eti_sel

Function name	syscfg_timer_eti_sel
Function prototype	void syscfg_timer_eti_sel(syscfg_timersel_enum timer_num, uint32_t eti_num);
Function descriptions	select TIMER external trigger source
Precondition	-
The called functions	-
Input parameter{in}	
timer_num	specify TIMER
TIMER0SEL	select TIMER0
TIMER7SEL	select TIMER7
TIMER19SEL	select TIMER19
TIMER20SEL	select TIMER20
Input parameter{in}	
etr_num	specify external trigger source
TIMER_ETI_TRGx	TIMER external trigger source x, x = 0,1,2,3
TIMER_ETI_TRG_NON	do not select TIMER external trigger source

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

Example:

```
/* select TIMER0 external trigger source 0 */
```

```
syscfg_timer_eti_sel (TIMER0SEL, TIMER_ETI_TRG0);
```

syscfg_timer_bkin_select_trigsel

The description of syscfg_timer_bkin_select_trigsel is shown as below:

Table 3-652. Function syscfg_timer_bkin_select_trigsel

Function name	syscfg_timer_bkin_select_trigsel
Function prototype	void syscfg_timer_bkin_select_trigsel(uint32_t bkin_source);
Function descriptions	select TRIGSEL as TIMER break input source
Precondition	-
The called functions	-
Input parameter{in}	
bkin_source	specify TIMER break input source
<i>TIMERx_BKINy_TRIG</i>	TIMERx break input y source select from TRIGSEL, x=0,7,19,20 y=0,1,2,3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TRIGSEL as TIMER0 break input source 0 */
```

```
syscfg_timer_bkin_select_trigsel(TIMER0_BKIN0_TRIG);
```

syscfg_timer_bkin_select_gpio

The description of syscfg_timer_bkin_select_gpio is shown as below:

Table 3-653. Function syscfg_timer_bkin_select_gpio

Function name	syscfg_timer_bkin_select_gpio
Function prototype	void syscfg_timer_bkin_select_gpio(uint32_t bkin_source);
Function descriptions	select GPIO as TIMER break input source
Precondition	-
The called functions	-
Input parameter{in}	

bkin_source	specify TIMER break input source
<i>TIMERx_BKINy_TRIG</i>	TIMERx break input y source select from TRIGSEL, x=0,7,19,20 y=0,1,2,3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select GPIO as TIMER0 break input source 0 */
```

```
syscfg_timer_bkin_select_gpio(TIMER0_BKIN0_TRIG);
```

syscfg_timer7_ch0n_select

The description of syscfg_timer7_ch0n_select is shown as below:

Table 3-654. Function syscfg_timer7_ch0n_select

Function name	syscfg_timer7_ch0n_select
Function prototype	void syscfg_timer7_ch0n_select(uint32_t timer7_ch0n_in);
Function descriptions	select TIMER7 channel0 complementary input source
Precondition	-
The called functions	-
Input parameter{in}	
timer7_ch0n_in	specify TIMER7 channel0 complementary input source
<i>TIMER7CH0N_TIMER7CH0_TIMER0CH0_IN</i>	select exclusive or of TIMER7_CH0_IN, TIMER7_CH0N_IN, and TIMER0_CH0_IN
<i>TIMER7_CH0N_IN</i>	select TIMER7_CH0N_IN
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER7 channel0 complementary input source */
```

```
syscfg_timer7_ch0n_select (TIMER7_CH0N_IN);
```

syscfg_lock_config

The description of syscfg_lock_config is shown as below:

Table 3-655. Function syscfg_lock_config

Function name	syscfg_lock_config
Function prototype	void syscfg_lock_config(uint32_t syscfg_lock);
Function descriptions	configure TIMER0/7/19/20 break input to the selected parameter connection

Precondition	-
The called functions	-
Input parameter{in}	
syscfg_lock	specify the parameter to be connected
<i>SYSCFG_LOCK_LOCKUP</i>	Cortex-M33 lockup output connected to the TIMER0/7/19/20 break input
<i>SYSCFG_LOCK_SRAM_ECC_ERROR</i>	SRAM ECC check error connected to the TIMER0/7/19/20 break input
<i>SYSCFG_LOCK_LVD</i>	LVD interrupt connected to the TIMER0/7/19/20 break input
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure Cortex-M33 lockup output connected to the break input */
syscfg_lock_config(SYSCFG_LOCK_LOCKUP);
```

syscfg_flag_get

The description of syscfg_flag_get is shown as below:

Table 3-656. Function syscfg_flag_get

Function name	syscfg_flag_get
Function prototype	FlagStatus syscfg_flag_get(uint32_t flag);
Function descriptions	get SYSCFG flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify the flag in SYSCFG_STAT to check
<i>SYSCFG_FLAG_SRAM_MECCMERR</i>	SRAM multi-bits non-correction ECC error flag
<i>SYSCFG_FLAG_SRAM_MECCSERR</i>	SRAM single bit correction ECC error flag
<i>SYSCFG_FLAG_FLASH_HECCERR</i>	FLASH ECC NMI error flag
<i>SYSCFG_FLAG_CKM_NMIERR</i>	HXTAL clock monitor NMI error flag
<i>SYSCFG_FLAG_NMIPI_NERR</i>	NMI pin error flag
Output parameter{out}	
-	-
Return value	

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

Example:

```
/* get SRAM multi-bits non-correction ECC error flag */
```

```
FlagStatus flag;
```

```
flag = syscfg_flag_get (SYSCFG_FLAG_SRAM_ECCMERR);
```

syscfg_flag_clear

The description of syscfg_flag_clear is shown as below:

Table 3-657. Function syscfg_flag_clear

Function name	syscfg_flag_clear
Function prototype	void syscfg_flag_clear(uint32_t flag);
Function descriptions	clear SYSCFG flags
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify the flag in SYSCFG_STAT to check
SYSCFG_FLAG_SRAM_ECCMERR	SRAM multi-bits non-correction ECC error flag
SYSCFG_FLAG_SRAM_ECCSERR	SRAM single bit correction ECC error flag
SYSCFG_FLAG_FLASH_HECCERR	FLASH ECC NMI error flag
SYSCFG_FLAG_HXTAL_CKM_NMIERR	HXTAL clock monitor NMI error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear SRAM multi-bits non-correction ECC error flag */
```

```
syscfg_flag_clear (SYSCFG_FLAG_SRAM_ECCMERR);
```

syscfg_interrupt_enable

The description of syscfg_interrupt_enable is shown as below:

Table 3-658. Function syscfg_interrupt_enable

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

Function descriptions	enable SYSCFG interrupts
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify the interrupt in SYSCFG_CFG3
<i>SYSCFG_INT_SRAM CCME</i>	SRAM multi-bits non-correction ECC error interrupt
<i>SYSCFG_INT_SRAM CCSE</i>	SRAM single bit correction ECC error interrupt
<i>SYSCFG_INT_FLASH ECCE</i>	FLASH ECC NMI error interrupt
<i>SYSCFG_INT_CKMN M</i>	HXTAL clock monitor NMI error interrupt
<i>SYSCFG_INT_NMIPIN</i>	NMI pin error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SRAM multi-bits non-correction ECC error interrupt */
```

```
syscfg_interrupt_enable (SYSCFG_INT_SRAMCCME);
```

syscfg_interrupt_disable

The description of syscfg_interrupt_disable is shown as below:

Table 3-659. Function syscfg_interrupt_disable

Function name	syscfg_interrupt_disable
Function prototype	void syscfg_interrupt_disable(uint32_t interrupt);
Function descriptions	disable SYSCFG interrupts
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify the interrupt in SYSCFG_CFG3
<i>SYSCFG_INT_SRAM CCME</i>	SRAM multi-bits non-correction ECC error interrupt
<i>SYSCFG_INT_SRAM CCSE</i>	SRAM single bit correction ECC error interrupt
<i>SYSCFG_INT_FLASH ECCE</i>	FLASH ECC NMI error interrupt
<i>SYSCFG_INT_CKMN M</i>	HXTAL clock monitor NMI error interrupt

<i>SYSCFG_INT_NMIPIN</i>	NMI pin error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SRAM multi-bits non-correction ECC error interrupt */
syscfg_interrupt_disable (SYSCFG_INT_SRAM_ECCME);
```

syscfg_interrupt_flag_get

The description of syscfg_interrupt_flag_get is shown as below:

Table 3-660. Function syscfg_interrupt_flag_get

Function name	syscfg_interrupt_flag_get
Function prototype	FlagStatus syscfg_interrupt_flag_get(uint32_t interrupt);
Function descriptions	get SYSCFG interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	specify the interrupt in SYSCFG_CFG3
<i>SYSCFG_INT_FLAG_SRAM_ECCMERR</i>	SRAM multi-bits non-correction ECC error interrupt flag
<i>SYSCFG_INT_FLAG_SRAM_ECCSERR</i>	SRAM single bit correction ECC error interrupt flag
<i>SYSCFG_INT_FLAG_FLASH_ECCERR</i>	FLASH ECC NMI error interrupt flag
<i>SYSCFG_INT_FLAG_HXTAL_CLK_MONIERR</i>	HXTAL clock monitor NMI error interrupt flag
<i>SYSCFG_INT_FLAG_NMIPINERR</i>	NMI pin error interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SRAM multi-bits non-correction ECC error interrupt flag */
FlagStatus flag;
flag = syscfg_interrupt_flag_get (SYSCFG_INT_FLAG_SRAM_ECCMERR);
```

syscfg_sram_ecc_address_get

The description of syscfg_sram_ecc_address_get is shown as below:

Table 3-661. Function syscfg_sram_ecc_address_get

Function name	syscfg_sram_ecc_address_get
Function prototype	uint16_t syscfg_sram_ecc_address_get(void);
Function descriptions	get the address where SRAM ECC error occur on
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	the address where SRAM ECC error occur on, 0x0000 – 0xFFFF

Example:

```
/* get the address where SRAM ECC error occur on */
```

```
uint16_t sram_ecc_addr;
```

```
sram_ecc_addr = syscfg_sram_ecc_address_get();
```

syscfg_sram_ecc_bit_get

The description of syscfg_sram_ecc_bit_get is shown as below:

Table 3-662. Function syscfg_sram_ecc_bit_get

Function name	syscfg_sram_ecc_bit_get
Function prototype	uint8_t syscfg_sram_ecc_bit_get(void);
Function descriptions	get the bit which has SRAM ECC signle error
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
Uint8_t	which bit has SRAM ECC signle error, 0x00 – 0xFF

Example:

```
/* get the bit which has SRAM ECC signle error */
```

```
uint8_t sram_ecc_bit;
```

```
sram_ecc_bit = syscfg_sram_ecc_bit_get();
```

3.22. 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 three sorts: advanced timer (TIMERx, x=0, 7, 19, 20), general level0 timer (TIMERx, x=1) and basic timer (TIMERx, x=5, 6). The specific functions of different types of timer are different. The TIMER registers are listed in chapter [3.22.1](#), the TIMER firmware functions are introduced in chapter [3.22.2](#).

3.22.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

Table 3-663. 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_MCHCTL0	TIMER multi mode channel control register 0
TIMER_MCHCTL1	TIMER multi mode channel control register 1
TIMER_MCHCTL2	TIMER multi mode channel control register 2
TIMER_IRMP	TIMER channel input remap register (only for TIMER1)
TIMER_MCH0CV	TIMER multi mode channel 0 capture or compare value register
TIMER_MCH1CV	TIMER multi mode channel 1 capture or compare value register
TIMER_MCH2CV	TIMER multi mode channel 2 capture or compare value register
TIMER_MCH3CV	TIMER multi mode channel 3 capture or compare value register

Registers	Descriptions
TIMER_CH0COMV_ADD	TIMER channel 0 additional compare value register
TIMER_CH1COMV_ADD	TIMER channel 1 additional compare value register
TIMER_CH2COMV_ADD	TIMER channel 2 additional compare value register
TIMER_CH3COMV_ADD	TIMER channel 3 additional compare value register
TIMER_CTL2	TIMER control register 2
TIMER_BRKCFG	TIMER break configuration register
TIMER_FCCHP0	TIMER free complementary channel protection register 0
TIMER_FCCHP1	TIMER free complementary channel protection register 1
TIMER_FCCHP2	TIMER free complementary channel protection register 2
TIMER_FCCHP3	TIMER free complementary channel protection register 3
TIMER_DMACFG	DMA configuration register
TIMER_DMATB	DMA transfer buffer register
TIMER_CFG	Configuration register

3.22.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

Table 3-664. TIMERx firmware function

Function name	Function description
timer_deinit	deinit a TIMER
timer_struct_para_init	initialize the parameters of TIMER init parameter struct with the default values
timer_init	initialize TIMER counter
timer_enable	enable a timer
timer_disable	disable a timer
timer_auto_reload_shadow_enable	enable the auto reload shadow function
timer_auto_reload_shadow_disable	disable the auto reload shadow function
timer_update_event_enable	enable the update event
timer_update_event_disable	disable the update event
timer_counter_alignment	set TIMER counter alignment mode
timer_counter_up_direction	set TIMER counter up direction
timer_counter_down_direction	set TIMER counter down direction
timer_prescaler_config	configure TIMER prescaler
timer_repetition_value_config	configure TIMER repetition register value
timer_autoreload_value_config	configure TIMER autoreload register value
timer_counter_value_config	configure TIMER counter register value
timer_counter_read	read TIMER counter value

Function name	Function description
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_channel_control_shadow_config	configure channel commutation control shadow register
timer_channel_control_shadow_update_config	configure TIMER channel control shadow register update control
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_output_struct_para_init	initialize the parameters of TIMER channel output parameter struct with the default values
timer_channel_output_config	configure TIMER channel output function
timer_channel_output_mode_config	configure TIMER channel output compare mode
timer_channel_output_pulse_value_config	configure TIMER channel output pulse value
timer_channel_output_shadow_config	configure TIMER channel output shadow function
timer_channel_output_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_config	configure TIMER channel input capture prescaler value

Function name	Function description
prescaler_config	
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_multi_mode_channel_output_parameter_struct_init	initialize TIMER multi mode channel output parameter struct
timer_multi_mode_channel_output_config	configure TIMER multi mode channel output function
timer_multi_mode_channel_mode_config	multi mode channel mode select
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_channel_remap_config	configure TIMER channel input remap function
timer_write_chxval_register_config	configure TIMER write CHxVAL register selection
timer_output_value_selection_config	configure TIMER output value selection
timer_output_match_pulse_select	configure TIMER output match pulse selection
timer_channel_composite_pwm_mode_config	configure the TIMER composite PWM mode
timer_channel_composite_pwm_mode_output_pulse_value_config	configure the TIMER composite PWM mode output pulse value
timer_channel_additional_compare_value_config	configure TIMER channel additional compare value
timer_channel_additional_output_shadow_config	configure TIMER channel additional output shadow function
timer_break_external_input_struct_init	initialize TIMER break external input parameter struct

Function name	Function description
para_init	
timer_break_external_input_config	configure TIMER break external input polarity
timer_break_external_input_enable	break external input enable
timer_break_external_input_disable	break external input disable
timer_break_external_input_polarity_config	configure TIMER break external input polarity
timer_channel_break_control_config	configure the TIMER channel break function
timer_channel_dead_time_config	configure the TIMER channel dead time function
timer_free_complementary_struct_para_init	initialize TIMER channel free complementary parameter struct with a default value
timer_channel_free_complementary_config	configure channel free complementary protection
timer_flag_get	get TIMER flags
timer_flag_clear	clear TIMER flags
timer_interrupt_enable	enable the TIMER interrupt
timer_interrupt_disable	disable the TIMER interrupt
timer_interrupt_flag_get	get timer interrupt flags
timer_interrupt_flag_clear	clear TIMER interrupt flags

Structure timer_parameter_struct

Table 3-665. 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 3-666. 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)

Member name	Function description
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 3-667. Structure timer_oc_parameter_struct

Member name	Function description
outputstate	channel output state(TIMER_CCX_ENABLE, TIMER_CCX_DISABLE)
outputnstate	channel complementary output state(TIMER_CCXN_ENABLE, TIMER_CCXN_DISABLE)
ocpolarity	channel output polarity(TIMER_OC_POLARITY_HIGH, TIMER_OC_POLARITY_LOW)
ocnpolarity	channel complementary output polarity(TIMER_OCN_POLARITY_HIGH, TIMER_OCN_POLARITY_LOW)
ocidlestate	idle state of channel output(TIMER_OC_IDLE_STATE_LOW, TIMER_OC_IDLE_STATE_HIGH)
ocnidlestate	idle state of channel complementary output(TIMER_OCN_IDLE_STATE_LOW, TIMER_OCN_IDLE_STATE_HIGH)

Structure timer_omc_parameter_struct

Table 3-668. Structure timer_omc_parameter_struct

Member name	Function description
outputmode	multi mode channel output mode selection(TIMER_MCH_MODE_INDEPENDENTLY, TIMER_MCH_MODE_MIRRORED, TIMER_MCH_MODE_COMPLEMENTARY)
outputstate	multi mode channel output state(TIMER_MCCX_ENABLE, TIMER_MCCX_DISABLE)
ocpolarity	multi mode channel output polarity(TIMER_OMC_POLARITY_HIGH, TIMER_OMC_POLARITY_LOW)

Structure timer_ic_parameter_struct

Table 3-669. 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)

Member name	Function description
icselection	channel input mode selection(TIMER_IC_SELECTION_DIRECTTI, TIMER_IC_SELECTION_INDIRECTTI, TIMER_IC_SELECTION_ITS, TIMER_IC_SELECTION_PAIR)
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)

Structure timer_break_ext_input_struct

Table 3-670. Structure timer_break_ext_input_struct

Member name	Function description
filter	break external input filter(0~15)
enable	break external input enable(ENABLE or DISABLE)
polarity	break external input polarity(TIMER_BRKIN_POLARITY_HIGH, TIMER_BRKIN_POLARITY_LOW)

Structure timer_free_complementary_parameter_struct

Table 3-671. Structure timer_free_complementary_parameter_struct

Member name	Function description
freecomstate	free complementary channel protection enable(TIMER_FCCHP_STATE_ENABLE, TIMER_FCCHP_STATE_DISABLE)
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)

timer_deinit

The description of timer_deinit is shown as below:

Table 3-672. Function timer_deinit

Function name	timer_deinit
Function prototype	void timer_deinit(uint32_t timer_periph);
Function descriptions	deinit a TIMER
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,1,5,6,7,19,20)	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 3-673. 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 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 3-674. 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</i> (<i>x</i> =0,1,5,6,7,19,20)	TIMER peripheral selection
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to 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 3-675. 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</i> (<i>x</i> =0,1,5,6,7,19,20)	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 3-676. 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
<i>TIMERx</i> (x=0,1,5,6,7,19,20)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 */
timer_disable(TIMER0);
```

timer_auto_reload_shadow_enable

The description of timer_auto_reload_shadow_enable is shown as below:

Table 3-677. 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
<i>TIMERx</i> (x=0,1,5,6,7,19,20)	TIMER peripheral selection

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMER0 auto reload shadow function */
timer_auto_reload_shadow_enable(TIMER0);
```

timer_auto_reload_shadow_disable

The description of timer_auto_reload_shadow_disable is shown as below:

Table 3-678. Function timer_auto_reload_shadow_disable

Function name	timer_auto_reload_shadow_disable
Function prototype	void timer_auto_reload_shadow_disable (uint32_t timer_periph);
Function descriptions	disable the auto reload shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,1,5,6,7,19,20)	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 3-679. 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</i> (<i>x</i> =0,1,5,6,7,19,20)	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 3-680. 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</i> (<i>x</i> =0,1,5,6,7,19,20)	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 3-681. 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=0,1,7,19,20)</i>	TIMER peripheral selection
Input parameter{in}	
aligned	alignment mode
<i>TIMER_COUNTER_EDGE</i>	edge-aligned mode
<i>TIMER_COUNTER_CENTER_DOWN</i>	center-aligned and counting down assert mode
<i>TIMER_COUNTER_CENTER_UP</i>	center-aligned and counting up assert mode
<i>TIMER_COUNTER_CENTER_BOTH</i>	center-aligned and counting up/down assert mode
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 3-682. 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 TIMERx counter no center-aligned mode (edge-aligned mode)
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,1,7,19,20)</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 3-683. 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 TIMEx counter no center-aligned mode (edge-aligned mode)
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMEx (x=0,1,7,19,20)</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 3-684. 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>TIMEx(x=0,1,5,6,7,19,20)</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 3-685. 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=0,7,19,20)</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 3-686. Function timer_autoreload_value_config

Function name	timer_autoreload_value_config
Function prototype	void timer_autoreload_value_config(uint32_t timer_periph, uint16_t autoreload);
Function descriptions	configure TIMER autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,1,5,6,7,19,20)	TIMER peripheral selection
Input parameter{in}	
autoreload	the counter auto-reload value (0~65535)
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 3-687. Function timer_counter_value_config

Function name	timer_counter_value_config
Function prototype	void timer_counter_value_config(uint32_t timer_periph, uint16_t counter);
Function descriptions	configure TIMER counter register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,1,5,6,7,19,20)	TIMER peripheral selection
Input parameter{in}	
counter	the counter value (0~65535)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 counter register value */
```

```
timer_counter_value_config(TIMER0, 999);
```

timer_counter_read

The description of timer_counter_read is shown as below:

Table 3-688. 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, 1, 5, 6, 7, 19, 20)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint32_t	counter value(0~65535)

Example:

```
/* read TIMER0 counter value */
```

```
uint32_t i = 0;
```

```
i = timer_counter_read(TIMER0);
```

timer_prescaler_read

The description of timer_prescaler_read is shown as below:

Table 3-689. 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
<i>TIMERx(x=0, 1, 5, 6, 7, 19, 20)</i>	TIMER peripheral selection
Output parameter{out}	
-	-

Return value	
uint16_t	prescaler register value (0~65535)

Example:

```
/* read TIMER0 prescaler value */
```

```
uint16_t i = 0;
```

```
i = timer_prescaler_read(TIMER0);
```

timer_single_pulse_mode_config

The description of timer_single_pulse_mode_config is shown as below:

Table 3-690. 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, uint32_t spmode);
Function descriptions	configure TIMER single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0, 1, 5, 6, 7, 19, 20)</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 3-691. 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</i> (<i>x</i> =0, 1, 5, 6, 7, 19, 20)	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_channel_control_shadow_config

The description of timer_channel_control_shadow_config is shown as below:

Table 3-692. 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	configure channel commutation control shadow register
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0, 7, 19, 20)	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 commutation 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 3-693. 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 TIMER channel control shadow register update control
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</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_dma_enable

The description of timer_dma_enable is shown as below:

Table 3-694. Function timer_dma_enable

Function name	timer_dma_enable
Function prototype	void timer_dma_enable(uint32_t timer_periph, uint32_t dma);
Function descriptions	enable the TIMER DMA
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,1,5,6,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
dma	timer DMA source enable
<i>TIMER_DMA_UPD</i>	update DMA request, <i>TIMERx(x=0,1,5,6,7,19,20)</i>
<i>TIMER_DMA_CH0D</i>	channel 0 DMA request, <i>TIMERx(x=0,1,7,19,20)</i>
<i>TIMER_DMA_CH1D</i>	channel 1 DMA request, <i>TIMERx(x=0,1,7,19,20)</i>
<i>TIMER_DMA_CH2D</i>	channel 2 DMA request, <i>TIMERx(x=0,1,7,19,20)</i>
<i>TIMER_DMA_CH3D</i>	channel 3 DMA request, <i>TIMERx(x=0,1,7,19,20)</i>
<i>TIMER_DMA_CMTD</i>	commutation DMA request, <i>TIMERx(x=0,7,19,20)</i>
<i>TIMER_DMA_TRGD</i>	trigger DMA request, <i>TIMERx(x=0,1,7,19,20)</i>
<i>TIMER_DMA_MCH0D</i>	multi mode channel 0 DMA request, <i>TIMERx(x=0,7,19,20)</i>
<i>TIMER_DMA_MCH1D</i>	multi mode channel 1 DMA request, <i>TIMERx(x=0,7,19,20)</i>
<i>TIMER_DMA_MCH2D</i>	multi mode channel 2 DMA request, <i>TIMERx(x=0,7,19,20)</i>
<i>TIMER_DMA_MCH3D</i>	multi mode channel 3 DMA request, <i>TIMERx(x=0,7,19,20)</i>
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 3-695. Function timer_dma_disable

Function name	timer_dma_disable
Function prototype	void timer_dma_disable (uint32_t timer_periph, uint32_t dma);
Function descriptions	disable the TIMER DMA
Precondition	-
The called functions	-
Input parameter{in}	

timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0, 1, 5, 6, 7, 19, 20)	please refer to the following parameters
Input parameter{in}	
dma	timer DMA source disable
<i>TIMER_DMA_UPD</i>	update DMA request, <i>TIMERx</i> (<i>x</i> =0, 1, 5, 6, 7, 19, 20)
<i>TIMER_DMA_CH0D</i>	channel 0 DMA request, <i>TIMERx</i> (<i>x</i> =0, 1, 7, 19, 20)
<i>TIMER_DMA_CH1D</i>	channel 1 DMA request, <i>TIMERx</i> (<i>x</i> =0, 1, 7, 19, 20)
<i>TIMER_DMA_CH2D</i>	channel 2 DMA request, <i>TIMERx</i> (<i>x</i> =0, 1, 7, 19, 20)
<i>TIMER_DMA_CH3D</i>	channel 3 DMA request, <i>TIMERx</i> (<i>x</i> =0, 1, 7, 19, 20)
<i>TIMER_DMA_CMTD</i>	commutation DMA request, <i>TIMERx</i> (<i>x</i> =0, 7, 19, 20)
<i>TIMER_DMA_TRGD</i>	trigger DMA request, <i>TIMERx</i> (<i>x</i> =0, 1, 7, 19, 20)
<i>TIMER_DMA_MCH0D</i>	multi mode channel 0 DMA request, <i>TIMERx</i> (<i>x</i> =0, 7, 19, 20)
<i>TIMER_DMA_MCH1D</i>	multi mode channel 1 DMA request, <i>TIMERx</i> (<i>x</i> =0, 7, 19, 20)
<i>TIMER_DMA_MCH2D</i>	multi mode channel 2 DMA request, <i>TIMERx</i> (<i>x</i> =0, 7, 19, 20)
<i>TIMER_DMA_MCH3D</i>	multi mode channel 3 DMA request, <i>TIMERx</i> (<i>x</i> =0, 7, 19, 20)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMER0 update DMA */
```

```
timer_dma_disable(TIMER0, TIMER_DMA_UPD);
```

timer_channel_dma_request_source_select

The description of timer_channel_dma_request_source_select is shown as below:

Table 3-696. 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</i> (<i>x</i> =0, 1, 5, 6, 7, 19, 20)	TIMER peripheral selection
Input parameter{in}	
dma_request	channel DMA request source selection
<i>TIMER_DMAREQUEST</i>	DMA request of channel <i>n</i> is sent when channel <i>y</i> event occurs

<i>_CHANNELEVENT</i>	
<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 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 3-697. 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=0,1,7,19,20)
<i>TIMER_DMACFG_DMA_TA_CTL1</i>	DMA transfer address is TIMER_CTL1, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA_TA_SMCFG</i>	DMA transfer address is TIMER_SMCFG, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA_TA_DMAINTEN</i>	DMA transfer address is TIMER_DMAINTEN, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA_TA_INTF</i>	DMA transfer address is TIMER_INTF, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA_TA_SWEVG</i>	DMA transfer address is TIMER_SWEVG, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA_TA_CHCTL0</i>	DMA transfer address is TIMER_CHCTL0, TIMERx(x=0,1,7,19,20)

<i>TIMER_DMACFG_DMA TA_CHCTL1</i>	DMA transfer address is TIMER_CHCTL1, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_CHCTL2</i>	DMA transfer address is TIMER_CHCTL2, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_CNT</i>	DMA transfer address is TIMER_CNT, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_PSC</i>	DMA transfer address is TIMER_PSC, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_CAR</i>	MA transfer address is TIMER_CAR, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_CREP</i>	DMA transfer address is TIMER_CREP, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_CH0CV</i>	DMA transfer address is TIMER_CH0CV, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_CH1CV</i>	DMA transfer address is TIMER_CH1CV, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_CH2CV</i>	DMA transfer address is TIMER_CH2CV, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_CH3CV</i>	DMA transfer address is TIMER_CH3CV, TIMERx(x=0,1,7,19,20)
<i>TIMER_DMACFG_DMA TA_CCHP</i>	DMA transfer address is TIMER_CCHP, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_MCHCTL0</i>	DMA transfer address is MCHCTL0, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_MCHCTL1</i>	DMA transfer address is MCHCTL1, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_MCHCTL2</i>	DMA transfer address is MCHCTL2, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_MCH0CV</i>	DMA transfer address is MCH0CV, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_MCH1CV</i>	DMA transfer address is MCH1CV, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_MCH2CV</i>	DMA transfer address is MCH2CV, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_MCH3CV</i>	DMA transfer address is MCH3CV, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_CH0COMV_ADD</i>	DMA transfer address is CH0COMV_ADD, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_CH1COMV_ADD</i>	DMA transfer address is CH1COMV_ADD, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA TA_CH2COMV_ADD</i>	DMA transfer address is CH2COMV_ADD, TIMERx(x=0,7,19,20)
<i>TIMER_DMACFG_DMA</i>	DMA transfer address is CH3COMV_ADD, TIMERx(x=0,7,19,20)

<i>TA_CH3COMV_ADD</i>	
<i>TIMER_DMACFG_DMA TA_CTL2</i>	DMA transfer address is CTL2, $TIMERx(x=0,7,19,20)$
<i>TIMER_DMACFG_DMA TA_BRKCFG</i>	DMA transfer address is BRKCFG, $TIMERx(x=0,7,19,20)$
<i>TIMER_DMACFG_DMA TA_FCCHP0</i>	DMA transfer address is FCCHP0, $TIMERx(x=0,7,19,20)$
<i>TIMER_DMACFG_DMA TA_FCCHP1</i>	DMA transfer address is FCCHP1, $TIMERx(x=0,7,19,20)$
<i>TIMER_DMACFG_DMA TA_FCCHP2</i>	DMA transfer address is FCCHP2, $TIMERx(x=0,7,19,20)$
<i>TIMER_DMACFG_DMA TA_FCCHP3</i>	DMA transfer address is FCCHP3, $TIMERx(x=0,7,19,20)$
Input parameter{in}	
dma_lenth	DMA transfer count
<i>TIMER_DMACFG_DMA TC_xTRANSFER</i>	($x=1\sim35$), DMA transfer x time
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TIMER0 DMA transfer */
```

```
timer_dma_transfer_config(TIMER0,                                TIMER_DMACFG_DMATA_CTL0,
TIMER_DMACFG_DMATC_5TRANSFER);
```

timer_event_software_generate

The description of timer_event_software_generate is shown as below:

Table 3-698. Function timer_event_software_generate

Function name	timer_event_software_generate
Function prototype	void timer_event_software_generate(uint32_t timer_periph, uint32_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, $TIMERx(x=0,1,5,6,7,19,20)$
<i>TIMER_EVENT_SRC_CH0G</i>	channel 0 capture or compare event generation, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_EVENT_SRC_CH1G</i>	channel 1 capture or compare event generation, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_EVENT_SRC_CH2G</i>	channel 2 capture or compare event generation, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_EVENT_SRC_CH3G</i>	channel 3 capture or compare event generation, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_EVENT_SRC_MTG</i>	channel commutation event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_TRG</i>	trigger event generation, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_EVENT_SRC_BRK</i>	break event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_MCH0G</i>	multi mode channel 0 capture or compare event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_MCH1G</i>	multi mode channel 1 capture or compare event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_MCH2G</i>	multi mode channel 2 capture or compare event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_MCH3G</i>	multi mode channel 3 capture or compare event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_CH0COMADDG</i>	channel 0 additional compare event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_CH1COMADDG</i>	channel 1 additional compare event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_CH2COMADDG</i>	channel 2 additional compare event generation, $TIMERx(x=0,7,19,20)$
<i>TIMER_EVENT_SRC_CH3COMADDG</i>	channel 3 additional compare event generation, $TIMERx(x=0,7,19,20)$
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 3-699. 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 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 3-700. 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=0,7,19,20)</i>	TIMER peripheral selection
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to Structure timer break parameter struct.
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure TIMER0 break function */

timer_break_parameter_struct timer_breakpara;

timer_breakpara.runoffstate      = TIMER_ROS_STATE_DISABLE;
timer_breakpara.ideloffstate     = TIMER_IOS_STATE_DISABLE ;
timer_breakpara.deadtime         = 255;
timer_breakpara.breakpolarity    = TIMER_BREAK_POLARITY_LOW;
timer_breakpara.outputautostate  = TIMER_OUTAUTO_ENABLE;
timer_breakpara.protectmode      = TIMER_CCHP_PROT_0;
timer_breakpara.breakstate       = TIMER_BREAK_ENABLE;

timer_break_config(TIMER0, &timer_breakpara);
```

timer_break_enable

The description of timer_break_enable is shown as below:

Table 3-701. 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
<i>TIMERx(x=0,7,19,20)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 break function*/

timer_break_enable (TIMER0);
```

timer_break_disable

The description of timer_break_disable is shown as below:

Table 3-702. 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=0,7,19,20)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMERO break function*/
```

```
timer_break_disable(TIMERO);
```

timer_automatic_output_enable

The description of timer_automatic_output_enable is shown as below:

Table 3-703. 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=0,7,19,20)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMERO output automatic function */
```

```
timer_automatic_output_enable(TIMER0);
```

timer_automatic_output_disable

The description of timer_automatic_output_disable is shown as below:

Table 3-704. 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=0,7,19,20)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 output automatic function */
```

```
timer_automatic_output_disable(TIMER0);
```

timer_primary_output_config

The description of timer_primary_output_config is shown as below:

Table 3-705. 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=0,7,19,20)</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_output_struct_para_init

The description of timer_channel_output_struct_para_init is shown as below:

Table 3-706. 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}	
ocpara	TIMER channel output parameter struct, the structure members can refer to 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 3-707. 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(x=0,1,7,19,20)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
TIMER_CH_3	IMER channel 3 (TIMERx(x=0,1,7,19,20))
Input parameter{in}	
ocpara	TIMER channel output parameter struct, the structure members can refer to 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 3-708. 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(x=0,1,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,7,19,20))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
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 3-709. 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>TIMERx(x=0,1,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,1,7,19,20))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
Input parameter{in}	
pulse	channel output pulse value(0~65535)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output pulse value */
```

```
timer_channel_output_pulse_value_config(TIMER0, TIMER_CH_0, 399);
```

timer_channel_output_shadow_config

The description of timer_channel_output_shadow_config is shown as below:

Table 3-710. 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> (<i>x</i> =0,1,7,19,20)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (<i>TIMERx</i> (<i>x</i> =0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (<i>TIMERx</i> (<i>x</i> =0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (<i>TIMERx</i> (<i>x</i> =0,1,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (<i>TIMERx</i> (<i>x</i> =0,1,7,19,20))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(<i>TIMERx</i> (<i>x</i> =0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(<i>TIMERx</i> (<i>x</i> =0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(<i>TIMERx</i> (<i>x</i> =0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(<i>TIMERx</i> (<i>x</i> =0,7,19,20))
Input parameter{in}	
ocshadow	channel output compare shadow
<i>TIMER_OC_SHADOW_ENABLE</i>	channel output compare shadow enable
<i>TIMER_OC_SHADOW_DISABLE</i>	channel output compare shadow disable
<i>TIMER_OMC_SHADOW_ENABLE</i>	multi mode channel output compare shadow enable
<i>TIMER_OMC_SHADOW_DISABLE</i>	multi mode channel output compare shadow 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_clear_config

The description of timer_channel_output_clear_config is shown as below:

Table 3-711. Function timer_channel_output_clear_config

Function name	timer_channel_output_clear_config
Function prototype	void timer_channel_output_clear_config(uint32_t timer_periph, uint16_t channel, uint16_t occlear);
Function descriptions	configure TIMER channel output clear function
Precondition	-
The called functions	-

Input parameter{in}	
timer_periph	TIMER periphera
<i>TIMERx(x=0,1,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,1,7,19,20))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
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
<i>TIMER_OMC_CLEAR_ENABLE</i>	multi mode channel output clear function enable
<i>TIMER_OMC_CLEAR_DISABLE</i>	multi mode 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 3-712. 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>TIMERx(x=0,1,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,1,7,19,20))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
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
<i>TIMER_OMC_POLARITY_HIGH</i>	multi mode channel output polarity is high
<i>TIMER_OMC_POLARITY_LOW</i>	multi mode 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 3-713. Function timer_channel_complementary_output_polarity_config

Function name	timer_channel_complementary_output_polarity_config
Function prototype	void timer_channel_complementary_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnpolarity);
Function descriptions	configure TIMER channel complementary output polarity

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2
<i>TIMER_CH_3</i>	TIMER channel 3
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 3-714. 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	TIMER channel

<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,1,7,19,20))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
Input parameter{in}	
state	TIMER channel enable state
<i>TIMER_CCX_ENABLE</i>	channel enable
<i>TIMER_CCX_DISABLE</i>	channel disable
<i>TIMER_MCCX_ENABLE</i>	multi mode channel enable
<i>TIMER_MCCX_DISABLE</i>	multi mode 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 3-715. 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(x=0,7,19,20)</i>	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2

<i>TIMER_CH_3</i>	TIMER channel 3
Input parameter{in}	
state	TIMER channel complementary output enable state
<i>TIMER_CCXN_ENABLE</i>	channel complementary enable
<i>TIMER_CCXN_DISABL</i> <i>E</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 3-716. 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 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 3-717. 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	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,1,7,19,20))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
Input parameter{in}	
icpara	TIMER channel input parameter struct, the structure members can refer to 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 3-718. 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	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,1,7,19,20))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
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 3-719. 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
TIMEx	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMEx(x=0,1,7,19,20))
TIMER_CH_1	TIMER channel 1 (TIMEx(x=0,1,7,19,20))
TIMER_CH_2	TIMER channel 2 (TIMEx(x=0,1,7,19,20))
TIMER_CH_3	TIMER channel 3 (TIMEx(x=0,1,7,19,20))
TIMER_MCH_0	TIMER multi mode channel 0(TIMEx(x=0,7,19,20))
TIMER_MCH_1	TIMER multi mode channel 1(TIMEx(x=0,7,19,20))
TIMER_MCH_2	TIMER multi mode channel 2(TIMEx(x=0,7,19,20))
TIMER_MCH_3	TIMER multi mode channel 3(TIMEx(x=0,7,19,20))
Output parameter{out}	
-	-
Return value	
uint32_t	channel capture compare register value (0~65535)

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 3-720. 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</i> (<i>x</i> =0, 1, 7, 19, 20)	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
<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 Structure timer_ic_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 input pwm capture parameter */
timer_ic_parameter_struct timer_ic_initpara;

timer_ic_initpara.icpolarity = TIMER_IC_POLARITY_RISING;

timer_ic_initpara.icselection = TIMER_IC_SELECTION_DIRECTTTI;

timer_ic_initpara.icprescaler = TIMER_IC_PSC_DIV1;

timer_ic_initpara.icfilter = 0x0;

timer_input_pwm_capture_config(TIMER0, TIMER_CH_0, &timer_ic_initpara);

```

timer_hall_mode_config

The description of timer_hall_mode_config is shown as below:

Table 3-721. 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
<i>TIMERx</i> (<i>x</i> =0, 1, 7, 19, 20)	TIMER peripheral selection
Input parameter{in}	
hallmode	TIMER hall sensor mode state
<i>TIMER_HALLINTERFA</i> <i>CE_ENABLE</i>	TIMER hall sensor mode enable
<i>TIMER_HALLINTERFA</i>	TIMER hall sensor mode disable

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

Example:

```
/* configure TIMER0 hall sensor mode */
```

```
timer_hall_mode_config (TIMER0, TIMER_HALLINTERFACE_ENABLE);
```

timer_multi_mode_channel_output_parameter_struct_init

The description of timer_multi_mode_channel_output_parameter_struct_init is shown as below:

Table 3-722. Function timer_multi_mode_channel_output_parameter_struct_init

Function name	timer_multi_mode_channel_output_parameter_struct_init
Function prototype	void timer_multi_mode_channel_output_parameter_struct_init(timer_omc_parameter_struct *omcpara);
Function descriptions	initialize TIMER multi mode channel output parameter struct
Precondition	-
The called functions	-
Input parameter{in}	
omcpara	TIMER multi mode channel output parameter struct, the structure members can refer to Structure timer_omc_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER multi mode channel output parameter struct with a default value */
```

```
timer_omc_parameter_struct timer_omcinitpara;
```

```
timer_multi_mode_channel_output_parameter_struct_init(&timer_omcinitpara);
```

timer_multi_mode_channel_output_config

The description of timer_multi_mode_channel_output_config is shown as below:

Table 3-723. Function timer_multi_mode_channel_output_config

Function name	timer_multi_mode_channel_output_config
Function prototype	void timer_multi_mode_channel_output_config(uint32_t timer_periph,

	uint16_t channel, timer_omc_parameter_struct *omcpara);
Function descriptions	configure TIMER multi mode channel output function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
Input parameter{in}	
omcpara	TIMER multi mode channel output parameter struct, the structure members can refer to Structure timer_omc_parameter_struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 multi mode channel 0 output function */
```

```
timer_omc_parameter_struct timer_omcinitpara;
```

```
omcpara->outputmode = TIMER_MCH_MODE_INDEPENDENTLY;
```

```
omcpara->outputstate = TIMER_MCCX_ENABLE;
```

```
omcpara->ocpolarity = TIMER_OMC_POLARITY_HIGH;
```

```
timer_multi_mode_channel_output_parameter_struct_init(TIMER0,      TIMER_MCH_0,
&timer_omcinitpara);
```

timer_multi_mode_channel_mode_config

The description of timer_multi_mode_channel_mode_config is shown as below:

Table 3-724. Function timer_multi_mode_channel_mode_config

Function name	timer_multi_mode_channel_mode_config
Function prototype	void timer_multi_mode_channel_mode_config(uint32_t timer_periph, uint32_t channel, uint32_t multi_mode_sel);
Function descriptions	multi mode channel mode select
Precondition	-
The called functions	-

Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19,20))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19,20))
Input parameter{in}	
multi_mode_sel	multi mode channel mode selection
<i>TIMER_MCH_MODE_INDEPENDENTLY</i>	multi mode channel work in independently mode
<i>TIMER_MCH_MODE_MIRRORED</i>	multi mode channel work in mirrored output mode
<i>TIMER_MCH_MODE_COMPLEMENTARY</i>	multi mode channel work in complementary output mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER0 multi mode channel 0 mode */
```

```
timer_multi_mode_channel_mode_config(TIMER0, TIMER_MCH_0,
TIMER_MCH_MODE_INDEPENDENTLY);
```

timer_input_trigger_source_select

The description of timer_input_trigger_source_select is shown as below:

Table 3-725. 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=0,1,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
intrigger	input trigger source

<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI0</i>	Internal trigger input 0 (ITI0, TIMEx(x=0,1,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI1</i>	Internal trigger input 1 (ITI1, TIMEx(x=0,1,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI2</i>	Internal trigger input 2 (ITI2, TIMEx(x=0,1,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI3</i>	Internal trigger input 3 (ITI3, TIMEx(x=0,1,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOF_ED</i>	CIO edge flag (CIOF_ED, TIMEx(x=0,1,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOFE0</i>	channel 0 input Filtered output (CIOFE0, TIMEx(x=0,1,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_C1FE1</i>	channel 1 input Filtered output (C1FE1, TIMEx(x=0,1,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ETIFP</i>	external trigger input filter output(ETIFP, TIMEx(x=0,1,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI2FE2</i>	filtered channel 2 input(TIMEx(x=0,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI3FE3</i>	filtered channel 3 input(TIMEx(x=0,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_MCI0FEM0</i>	filtered multi mode channel 0 input(TIMEx(x=0,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_MCI1FEM1</i>	filtered multi mode channel 1 input(TIMEx(x=0,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_MCI2FEM2</i>	filtered multi mode channel 2 input(TIMEx(x=0,7,19,20))
<i>TIMER_SMCFG_TRGS</i> <i>EL_MCI3FEM3</i>	filtered multi mode channel 3 input(TIMEx(x=0,7,19,20))
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 3-726. 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</i> (x=0,1,5,6,7,19,20)	TIMER peripheral selection
Input parameter{in}	
outtrigger	trigger output source
<i>TIMER_TRI_OUT_SRC_RESET</i>	the UPG bit as trigger output(TIMERx(x=0,1,5,6,7,19,20))
<i>TIMER_TRI_OUT_SRC_ENABLE</i>	the counter enable signal TIMER_CTL0_CEN as trigger output(TIMERx(x=0,1,5,6,7,19,20))
<i>TIMER_TRI_OUT_SRC_UPDATE</i>	update event as trigger output(TIMERx(x=0,1,5,6,7,19,20))
<i>TIMER_TRI_OUT_SRC_CH0</i>	a capture or a compare match occurred in channel 0 as trigger output TRGO(TIMERx(x=0,1,7,19,20))
<i>TIMER_TRI_OUT_SRC_O0CPRE</i>	O0CPRE as trigger output(TIMERx(x=0,1,7,19,20))
<i>TIMER_TRI_OUT_SRC_O1CPRE</i>	O1CPRE as trigger output(TIMERx(x=0,1,7,19,20))
<i>TIMER_TRI_OUT_SRC_O2CPRE</i>	O2CPRE as trigger output(TIMERx(x=0,1,7,19,20))
<i>TIMER_TRI_OUT_SRC_O3CPRE</i>	O3CPRE as trigger output(TIMERx(x=0,1,7,19,20))
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 3-727. 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=0,1,7,19,20)</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 3-728. 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, uint32_t masterslave);
Function descriptions	configure TIMER master slave mode
Precondition	-

The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx (x=0,1,7,19,20)</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 3-729. 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=0,1,7,19,20)</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>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 3-730. 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, 1, 7, 19, 20)</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>FALLING</i>	
<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 3-731. 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=0,1,7,19,20)</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 3-732. 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=0,1,7,19,20)</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 1 (ITI1)
<i>TIMER_SMCFG_TRGS_EL_ITI2</i>	Internal trigger input 2 (ITI2)
<i>TIMER_SMCFG_TRGS_EL_ITI3</i>	Internal trigger input 3 (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 3-733. 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=0,1,7,19,20)</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_C11FE1</i>	channel 1 input filtered output (C11FE1)
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 3-734. 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</i> (x=0, 1, 7, 19, 20)	TIMER peripheral selection
Input parameter{in}	
extprescaler	ETI external trigger prescaler
<i>TIMER_EXT_TRI_PSC_</i> <i>OFF</i>	no divided
<i>TIMER_EXT_TRI_PSC_</i> <i>DIV2</i>	divided by 2
<i>TIMER_EXT_TRI_PSC_</i> <i>DIV4</i>	divided by 4
<i>TIMER_EXT_TRI_PSC_</i> <i>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 3-735. 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</i> (x=0, 1, 7, 19, 20)	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_external_clock_mode1_config(TIMER0,          TIMER_ETP_FALLING, 0);
```

timer_external_clock_mode1_disable

The description of timer_external_clock_mode1_disable is shown as below:

Table 3-736. 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=0,1,7,19,20)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 the external clock mode1 */
```

```
timer_external_clock_mode1_disable(TIMER0);
```

timer_channel_remap_config

The description of timer_channel_remap_config is shown as below:

Table 3-737. Function timer_channel_remap_config

Function name	timer_channel_remap_config
Function prototype	void timer_channel_remap_config(uint32_t timer_periph, uint32_t remap);
Function descriptions	configure TIMER channel remap function
Precondition	-
The called functions	timer_external_trigger_config
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i>	TIMER peripheral selection
Input parameter{in}	
remap	remap function selection
<i>TIMER1_CIO_RMP_GPIO</i> <i>O</i>	TIMER1 channel 0 input remap to GPIO pin
<i>TIMER1_CIO_RMP_LXT</i> <i>AL</i>	TIMER1 channel 0 input remap to LXTAL
<i>TIMER1_CIO_RMP_HXT</i> <i>AL</i>	TIMER1 channel 0 input remap to HXTAL/128
<i>TIMER1_CIO_RMP_CK</i> <i>OUT0SEL</i>	TIMER1 channel 0 input remap to CKOUT0SEL
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER1 channel 0 input is connected to GPIO */
```

```
timer_channel_remap_config(TIMER1, TIMER1_CIO_RMP_GPIO);
```

timer_write_chxval_register_config

The description of timer_write_chxval_register_config is shown as below:

Table 3-738. 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 ccsl);
Function descriptions	configure TIMER write CHxVAL register selection

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0, 1, 7, 19, 20)	TIMER peripheral selection
Input parameter{in}	
ccsel	write CHxVAL register selection
<i>TIMER_CHVSEL_DISABLE</i>	no effect
<i>TIMER_CHVSEL_ENABLE</i>	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 3-739. 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</i> (<i>x</i> =0, 7, 19, 20)	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_output_match_pulse_select

The description of timer_output_match_pulse_select is shown as below:

Table 3-740. Function timer_output_match_pulse_select

Function name	timer_output_match_pulse_select
Function prototype	void timer_output_match_pulse_select(uint32_t timer_periph, uint32_t channel, uint16_t pulsesel);
Function descriptions	configure TIMER output match pulse selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,7,19,20))
Input parameter{in}	
pulsesel	output match pulse selection
<i>TIMER_PULSE_OUTPUT_NORMAL</i>	channel output normal
<i>TIMER_PULSE_OUTPUT_CNT_UP</i>	pulse output only when counting up
<i>TIMER_PULSE_OUTPUT_CNT_DOWN</i>	pulse output only when counting down
<i>TIMER_PULSE_OUTPUT_CNT_BOTH</i>	pulse output when counting up or down
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 match pulse selection */
```

timer_output_match_pulse_select (TIMER0, TIMER_CH_0,
TIMER_PULSE_OUTPUT_CNT_UP);

timer_channel_composite_pwm_mode_config

The description of timer_channel_composite_pwm_mode_config is shown as below:

Table 3-741. Function timer_channel_composite_pwm_mode_config

Function name	timer_channel_composite_pwm_mode_config
Function prototype	void timer_channel_composite_pwm_mode_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
Function descriptions	configure the TIMER composite PWM mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,7,19,20))
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TIMER composite PWM mode */
```

```
timer_channel_composite_pwm_mode_config (TIMER0, TIMER_CH_0, ENABLE);
```

timer_channel_composite_pwm_mode_output_pulse_value_config

The description of timer_channel_composite_pwm_mode_output_pulse_value_config is shown as below:

Table 3-742. Function timer_channel_composite_pwm_mode_output_pulse_value_config

Function name	timer_channel_composite_pwm_mode_output_pulse_value_config
----------------------	--

Function prototype	void timer_channel_composite_pwm_mode_output_pulse_value_config(uint32_t timer_periph, uint32_t channel, uint32_t pulse, uint32_t add_pulse);
Function descriptions	configure the TIMER composite PWM mode output pulse value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,7,19,20))
Input parameter{in}	
pulse	channel compare value(0~65535)
Input parameter{in}	
add_pulse	channel additional compare value(0~65535)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output pulse value */
```

```
timer_channel_output_pulse_value_config(TIMER0, TIMER_CH_0, 399, 3999);
```

timer_channel_additional_compare_value_config

The description of timer_channel_additional_compare_value_config is shown as below:

Table 3-743. Function timer_channel_additional_compare_value_config

Function name	timer_channel_additional_compare_value_config
Function prototype	void timer_channel_additional_compare_value_config(uint32_t timer_periph, uint16_t channel, uint32_t value);
Function descriptions	configure TIMER channel additional compare value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	

channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,7,19,20))
<i>TIMER_CH_3</i>	IMER channel 3 (TIMERx(x=0,7,19,20))
Input parameter{in}	
value	channel additional compare value(0~65535)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 additional compare value */
```

```
timer_channel_additional_compare_value_config (TIMER0, TIMER_CH_0, 399);
```

timer_channel_additional_output_shadow_config

The description of timer_channel_additional_output_shadow_config is shown as below:

Table 3-744. Function timer_channel_additional_output_shadow_config

Function name	timer_channel_additional_output_shadow_config
Function prototype	void timer_channel_additional_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t aocshadow);
Function descriptions	configure TIMER channel additional output shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx (x=0,1,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,7,19,20))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,7,19,20))
Input parameter{in}	
aocshadow	channel additional output compare shadow
<i>TIMER_ADD_SHADOW_ENABLE</i>	channel additional output compare shadow enable
<i>TIMER_ADD_SHADOW_DISABLE</i>	channel additional output compare shadow disable
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/*configure TIMER0 channel 0 additional output shadow function */
```

```
timer_channel_additional_output_shadow_config      (TIMER0,          TIMER_CH_0,
TIMER_OC_SHADOW_ENABLE);
```

timer_break_external_input_struct_para_init

The description of timer_break_external_input_struct_para_init is shown as below:

Table 3-745. Function timer_break_external_input_struct_para_init

Function name	timer_break_external_input_struct_para_init
Function prototype	void timer_break_external_input_struct_para_init(timer_break_ext_input_struct *breakinpara);
Function descriptions	initialize TIMER break external input parameter struct
Precondition	-
The called functions	-
Input parameter{in}	
breakinpara	TIMER break external input parameter struct, the structure members can refer to Structure timer break ext input struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER break external input parameter struct with a default value */
```

```
timer_break_ext_input_struct breakinpara;
```

```
timer_break_external_input_struct_para_init (&breakinpara);
```

timer_break_external_input_config

The description of timer_break_external_input_config is shown as below:

Table 3-746. Function timer_break_external_input_config

Function name	timer_break_external_input_config
Function prototype	void timer_break_external_input_config(uint32_t timer_periph, uint32_t break_input, timer_break_ext_input_struct *breakinpara);
Function descriptions	configure break external input

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
break_input	break external input
<i>TIMER_BREAKINPUT_BRK0</i>	TIMER break external input 0
<i>TIMER_BREAKINPUT_BRK1</i>	TIMER break external input 1
<i>TIMER_BREAKINPUT_BRK2</i>	TIMER break external input 2
<i>TIMER_BREAKINPUT_BRK3</i>	TIMER break external input 3
Input parameter{in}	
breakinpara	TIMER break external input parameter struct, the structure members can refer to Structure timer break ext input struct .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 break external input */

timer_break_ext_input_struct timer_breakinpara;

timer_breakinpara.filter    = 15;

timer_breakinpara.enable = ENABLE;

timer_breakinpara.polarity = TIMER_BRKIN_POLARITY_HIGH;

timer_break_external_input_config(TIMER0,          TIMER_BREAKINPUT_BRK0,          &
timer_breakinpara);

```

timer_break_external_input_enable

The description of timer_break_external_input_enable is shown as below:

Table 3-747. Function timer_break_external_input_enable

Function name	timer_break_external_input_enable
Function prototype	void timer_break_external_input_enable(uint32_t timer_periph, uint32_t break_input);
Function descriptions	break external input enable

Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
break_input	break external input
<i>TIMER_BREAKINPUT_BRK0</i>	TIMER break external input 0
<i>TIMER_BREAKINPUT_BRK1</i>	TIMER break external input 1
<i>TIMER_BREAKINPUT_BRK2</i>	TIMER break external input 2
<i>TIMER_BREAKINPUT_BRK3</i>	TIMER break external input 3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 break external input */
```

```
timer_break_external_input_enable(TIMER0, TIMER_BREAKINPUT_BRK0);
```

timer_break_external_input_disable

The description of timer_break_external_input_disable is shown as below:

Table 3-748. Function timer_break_external_input_disable

Function name	timer_break_external_input_disable
Function prototype	void timer_break_external_input_disable(uint32_t timer_periph, uint32_t break_input);
Function descriptions	break external input disable
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
break_input	break external input
<i>TIMER_BREAKINPUT_BRK0</i>	TIMER break external input 0
<i>TIMER_BREAKINPUT_BRK1</i>	TIMER break external input 1

<i>BRK1</i>	
<i>TIMER_BREAKINPUT_</i> <i>BRK2</i>	TIMER break external input 2
<i>TIMER_BREAKINPUT_</i> <i>BRK3</i>	TIMER break external input 3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 break external input */
timer_break_external_input_disable (TIMER0, TIMER_BREAKINPUT_BRK0);
```

timer_break_external_input_polarity_config

The description of timer_break_external_input_polarity_config is shown as below:

Table 3-749. Function timer_break_external_input_polarity_config

Function name	timer_break_external_input_polarity_config
Function prototype	void timer_break_external_input_polarity_config(uint32_t timer_periph, uint32_t break_input, uint32_t polarity);
Function descriptions	configure TIMER break external input polarity
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx</i> (<i>x</i> =0,7,19,20)	please refer to the following parameters
Input parameter{in}	
break_input	break external input
<i>TIMER_BREAKINPUT_</i> <i>BRK0</i>	TIMER break external input 0
<i>TIMER_BREAKINPUT_</i> <i>BRK1</i>	TIMER break external input 1
<i>TIMER_BREAKINPUT_</i> <i>BRK2</i>	TIMER break external input 2
<i>TIMER_BREAKINPUT_</i> <i>BRK3</i>	TIMER break external input 3
Input parameter{in}	
polarity	break external input polarity
<i>TIMER_BRKIN_POLARI</i> <i>TY_HIGH</i>	break external input polarity is high
<i>TIMER_BRKIN_POLARI</i> <i>TY_LOW</i>	break external input polarity is low

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 break external input 0 polarity */
```

```
timer_break_external_input_polarity_config (TIMER0,    TIMER_BREAKINPUT_BRK0,
TIMER_BRKIN_POLARITY_HIGH);
```

timer_channel_break_control_config

The description of timer_channel_break_control_config is shown as below:

Table 3-750. Function timer_channel_break_control_config

Function name	timer_channel_break_control_config
Function prototype	void timer_channel_break_control_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
Function descriptions	configure the TIMER channel break function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2
<i>TIMER_CH_3</i>	TIMER channel 3
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 break function */
```

```
timer_channel_break_control_config (TIMER0, TIMER_CH_0, ENABLE);
```

timer_channel_dead_time_config

The description of timer_channel_dead_time_config is shown as below:

Table 3-751. Function timer_channel_dead_time_config

Function name	timer_channel_dead_time_config
Function prototype	void timer_channel_dead_time_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
Function descriptions	configure the TIMER channel free dead time function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2
<i>TIMER_CH_3</i>	TIMER channel 3
Input parameter{in}	
newvalue	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 free dead time function */
```

```
timer_channel_dead_time_config (TIMER0, TIMER_CH_0, ENABLE);
```

timer_free_complementary_struct_para_init

The description of timer_free_complementary_struct_para_init is shown as below:

Table 3-752. Function timer_free_complementary_struct_para_init

Function name	timer_free_complementary_struct_para_init
Function prototype	void timer_free_complementary_struct_para_init(timer_free_complementary_parameter_struct *freecompara);
Function descriptions	initialize TIMER channel free complementary parameter struct with a default value

Precondition	-
The called functions	-
Input parameter{in}	
freecompara	TIMER channel free complementary parameter struct, the structure members can refer to Structure timer_free_complementary_parameter_struct.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER channel free complementary parameter struct with a default value */
```

```
timer_free_complementary_parameter_struct timer_freecompara;
```

```
timer_free_complementary_struct_para_init (&timer_freecompara);
```

timer_channel_free_complementary_config

The description of timer_channel_free_complementary_config is shown as below:

Table 3-753. Function timer_channel_free_complementary_config

Function name	timer_channel_free_complementary_config
Function prototype	void timer_channel_free_complementary_config(uint32_t timer_periph, uint16_t channel, timer_free_complementary_parameter_struct *fcpa);
Function descriptions	configure channel free complementary protection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
<i>TIMERx(x=0,7,19,20)</i>	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2
<i>TIMER_CH_3</i>	TIMER channel 3
Input parameter{in}	
freecompara	TIMER channel free complementary parameter struct, the structure members can refer to Structure timer_free_complementary_parameter_struct.

Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize TIMER break parameter struct with a default value */

timer_free_complementary_parameter_struct timer_freecompara;

timer_freecompara.runoffstate = TIMER_FCCHP_STATE_ENABLE;

timer_freecompara.ideloffstate = TIMER_ROS_STATE_ENABLE;

timer_freecompara.deadtime = 255;

timer_freecompara.breakpolarity = TIMER_IOS_STATE_ENABLE;

timer_channel_free_complementary_config(&timer_freecompara);

```

timer_flag_get

The description of timer_flag_get is shown as below:

Table 3-754. 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> (x=0,1,5,6,7,19,20)	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
<i>TIMER_FLAG_UP</i>	update flag, TIMERx(x=0,1,5,6,7,19,20)
<i>TIMER_FLAG_CH0</i>	channel 0 capture or compare flag, TIMERx (x=0,1,7,19,20)
<i>TIMER_FLAG_CH1</i>	channel 1 capture or compare flag, TIMERx(x=0,1,7,19,20)
<i>TIMER_FLAG_CH2</i>	channel 2 capture or compare flag, TIMERx(x=0,1,7,19,20)
<i>TIMER_FLAG_CH3</i>	channel 3 capture or compare flag, TIMERx(x=0,1,7,19,20)
<i>TIMER_FLAG_CMT</i>	channel commutation flag, TIMERx(x=0,7,19,20)
<i>TIMER_FLAG_TRG</i>	trigger flag, TIMERx (x=0,1,7,19,20)
<i>TIMER_FLAG_BRK</i>	break flag, TIMERx(x=0,7,19,20)
<i>TIMER_FLAG_CH0O</i>	channel 0 overcapture flag, TIMERx(x=0,1,7,19,20)
<i>TIMER_FLAG_CH1O</i>	channel 1 overcapture flag, TIMERx(x=0,1,7,19,20)
<i>TIMER_FLAG_CH2O</i>	channel 2 overcapture flag, TIMERx(x=0,1,7,19,20)

<i>TIMER_FLAG_CH3O</i>	channel 3 overcapture flag, <i>TIMERx</i> (<i>x</i> =0,1,7,19,20)
<i>TIMER_FLAG_MCH0</i>	multi mode channel 0 capture or compare flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_MCH1</i>	multi mode channel 1 capture or compare flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_MCH2</i>	multi mode channel 2 capture or compare flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_MCH3</i>	multi mode channel 3 capture or compare flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_MCH0O</i>	multi mode channel 0 overcapture flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_MCH1O</i>	multi mode channel 1 overcapture flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_MCH2O</i>	multi mode channel 2 overcapture flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_MCH3O</i>	multi mode channel 3 overcapture flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_CH0CO</i> <i>MADD</i>	channel 0 additional compare flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_CH1CO</i> <i>MADD</i>	channel 1 additional compare flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_CH2CO</i> <i>MADD</i>	channel 2 additional compare flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_FLAG_CH3CO</i> <i>MADD</i>	channel 3 additional compare flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
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 3-755. 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, $TIMERx(x=0,1,5,6,7,19,20)$
<i>TIMER_FLAG_CH0</i>	channel 0 capture or compare flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_CH1</i>	channel 1 capture or compare flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_CH2</i>	channel 2 capture or compare flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_CH3</i>	channel 3 capture or compare flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_CMT</i>	channel commutation flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_TRG</i>	trigger flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_BRK</i>	break flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_CH0O</i>	channel 0 overcapture flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_CH1O</i>	channel 1 overcapture flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_CH2O</i>	channel 2 overcapture flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_CH3O</i>	channel 3 overcapture flag, $TIMERx(x=0,1,7,19,20)$
<i>TIMER_FLAG_MCH0</i>	multi mode channel 0 capture or compare flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_MCH1</i>	multi mode channel 1 capture or compare flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_MCH2</i>	multi mode channel 2 capture or compare flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_MCH3</i>	multi mode channel 3 capture or compare flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_MCH0O</i>	multi mode channel 0 overcapture flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_MCH1O</i>	multi mode channel 1 overcapture flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_MCH2O</i>	multi mode channel 2 overcapture flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_MCH3O</i>	multi mode channel 3 overcapture flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_CH0CO</i> <i>MADD</i>	channel 0 additional compare flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_CH1CO</i> <i>MADD</i>	channel 1 additional compare flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_CH2CO</i> <i>MADD</i>	channel 2 additional compare flag, $TIMERx(x=0,7,19,20)$
<i>TIMER_FLAG_CH3CO</i> <i>MADD</i>	channel 3 additional compare flag, $TIMERx(x=0,7,19,20)$
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 3-756. 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=0,1,5,6,7,19,20)
<i>TIMER_INT_CH0</i>	channel 0 capture or compare interrupt enable, TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_CH1</i>	channel 1 capture or compare interrupt enable, TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_CH2</i>	channel 2 capture or compare interrupt enable, TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_CH3</i>	channel 3 capture or compare interrupt enable , TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_CMT</i>	commutation interrupt enable, TIMERx (x=0,7,19,20)
<i>TIMER_INT_TRG</i>	trigger interrupt enable, TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_BRK</i>	break interrupt enable, TIMERx (x=0,7,19,20)
<i>TIMER_INT_MCH0</i>	multi mode channel 0 capture or compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_MCH1</i>	multi mode channel 1 capture or compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_MCH2</i>	multi mode channel 2 capture or compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_MCH3</i>	multi mode channel 3 capture or compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_CH0COMA DD</i>	channel 0 additional compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_CH0COMA DD</i>	channel 1 additional compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_CH0COMA DD</i>	channel 2 additional compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_CH0COMA DD</i>	channel 3 additional compare interrupt, TIMERx(x=0,7,19,20)
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 3-757. 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
<i>TIMERx</i>	please refer to the following parameters
Input parameter{in}	
interrupt	timer interrupt disable source
<i>TIMER_INT_UP</i>	update interrupt enable, TIMERx(x=0,1,5,6,7,19,20)
<i>TIMER_INT_CH0</i>	channel 0 capture or compare interrupt enable, TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_CH1</i>	channel 1 capture or compare interrupt enable, TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_CH2</i>	channel 2 capture or compare interrupt enable, TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_CH3</i>	channel 3 capture or compare interrupt enable , TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_CMT</i>	commutation interrupt enable, TIMERx (x=0,7,19,20)
<i>TIMER_INT_TRG</i>	trigger interrupt enable, TIMERx (x=0,1,7,19,20)
<i>TIMER_INT_BRK</i>	break interrupt enable, TIMERx (x=0,7,19,20)
<i>TIMER_INT_MCH0</i>	multi mode channel 0 capture or compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_MCH1</i>	multi mode channel 1 capture or compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_MCH2</i>	multi mode channel 2 capture or compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_MCH3</i>	multi mode channel 3 capture or compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_CH0COMA DD</i>	channel 0 additional compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_CH1COMA DD</i>	channel 1 additional compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_CH2COMA DD</i>	channel 2 additional compare interrupt, TIMERx(x=0,7,19,20)
<i>TIMER_INT_CH3COMA DD</i>	channel 3 additional compare interrupt, TIMERx(x=0,7,19,20)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMER0 update interrupt */
```

```
timer_interrupt_disable(TIMER0, TIMER_INT_UP);
```

timer_interrupt_flag_get

The description of timer_interrupt_flag_get is shown as below:

Table 3-758. 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 flags
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> (x=0,1,5,6,7,19,20)
<i>TIMER_INT_FLAG_CH0</i>	channel 0 capture or compare interrupt flag, <i>TIMERx</i> (x=0,1,7,19,20)
<i>TIMER_INT_FLAG_CH1</i>	channel 1 capture or compare interrupt flag, <i>TIMERx</i> (x=0,1,7,19,20)
<i>TIMER_INT_FLAG_CH2</i>	channel 2 capture or compare interrupt flag, <i>TIMERx</i> (x=0,1,7,19,20)
<i>TIMER_INT_FLAG_CH3</i>	channel 3 capture or compare interrupt flag, <i>TIMERx</i> (x=0,1,7,19,20)
<i>TIMER_INT_FLAG_CMT</i>	channel commutation interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_TRG</i>	trigger interrupt flag, <i>TIMERx</i> (x=0,1,7,19,20)
<i>TIMER_INT_FLAG_BRK</i>	break interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_MCH0</i>	multi mode channel 0 capture or compare interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_MCH1</i>	multi mode channel 1 capture or compare interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_MCH2</i>	multi mode channel 2 capture or compare interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_MCH3</i>	multi mode channel 3 capture or compare interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_CH0_COMADD</i>	channel 0 additional compare interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_CH1_COMADD</i>	channel 1 additional compare interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_CH2_COMADD</i>	channel 2 additional compare interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
<i>TIMER_INT_FLAG_CH3_COMADD</i>	channel 3 additional compare interrupt flag, <i>TIMERx</i> (x=0,7,19,20)
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 3-759. 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 flags
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> =0,1,5,6,7,19,20)
<i>TIMER_INT_FLAG_CH0</i>	channel 0 capture or compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,1,7,19,20)
<i>TIMER_INT_FLAG_CH1</i>	channel 1 capture or compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,1,7,19,20)
<i>TIMER_INT_FLAG_CH2</i>	channel 2 capture or compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,1,7,19,20)
<i>TIMER_INT_FLAG_CH3</i>	channel 3 capture or compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,1,7,19,20)
<i>TIMER_INT_FLAG_CMT</i>	channel commutation interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_TRG</i>	trigger interrupt flag, <i>TIMERx</i> (<i>x</i> =0,1,7,19,20)
<i>TIMER_INT_FLAG_BRK</i>	break interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_MCH0</i>	multi mode channel 0 capture or compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_MCH1</i>	multi mode channel 1 capture or compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_MCH2</i>	multi mode channel 2 capture or compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_MCH3</i>	multi mode channel 3 capture or compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_CH0_COMADD</i>	channel 0 additional compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_CH1_COMADD</i>	channel 1 additional compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_CH2_COMADD</i>	channel 2 additional compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)
<i>TIMER_INT_FLAG_CH3_COMADD</i>	channel 3 additional compare interrupt flag, <i>TIMERx</i> (<i>x</i> =0,7,19,20)

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

Example:

```
/* clear TIMER0 update interrupt flag */
```

```
timer_interrupt_flag_clear(TIMER0, TIMER_INT_FLAG_UP);
```

3.23. TRIGSEL

TRIGSEL is the trigger selection controller in the MCU. It allows software to select the trigger input signal for various peripherals. The TRIGSEL registers are listed in chapter [3.23.1](#), the TRIGSEL firmware functions are introduced in chapter [3.23.2](#).

3.23.1. Descriptions of Peripheral registers

TRIGSEL registers are listed in the table shown as below:

Table 3-760. TRIGSEL Registers

Registers	Descriptions
TRIGSEL_EXTOUT0	TRIGSEL trigger selection for EXTOUT0 register
TRIGSEL_EXTOUT1	TRIGSEL trigger selection for EXTOUT1 register
TRIGSEL_ADC0	TRIGSEL trigger selection for ADC0 register
TRIGSEL_ADC1	TRIGSEL trigger selection for ADC1 register
TRIGSEL_DAC	TRIGSEL trigger selection for DAC register
TRIGSEL_TIMER0IN	TRIGSEL trigger selection for TIMER0_ITI register
TRIGSEL_TIMER0BRKIN	TRIGSEL trigger selection for TIMER0_BRKIN register
TRIGSEL_TIMER7IN	TRIGSEL trigger selection for TIMER7_ITI register
TRIGSEL_TIMER7BRKIN	TRIGSEL trigger selection for TIMER7_BRKIN register
TRIGSEL_TIMER19IN	TRIGSEL trigger selection for TIMER19_ITI register
TRIGSEL_TIMER19BRKIN	TRIGSEL trigger selection for TIMER19_BRKIN register
TRIGSEL_TIMER20IN	TRIGSEL trigger selection for TIMER20_ITI register
TRIGSEL_TIMER20BRKIN	TRIGSEL trigger selection for TIMER20_BRKIN register
TRIGSEL_TIMER1IN	TRIGSEL trigger selection for TIMER1_ITI register
TRIGSEL_MFCOM	TRIGSEL trigger selection for MFCOM register
TRIGSEL_CAN0	TRIGSEL trigger selection for CAN0 register
TRIGSEL_CAN1	TRIGSEL trigger selection for CAN1 register

3.23.2. Descriptions of Peripheral functions

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

Table 3-761. TRIGSEL firmware function

Function name	Function description
trigsel_init	set the trigger input signal for target peripheral
trigsel_trigger_source_get	get the trigger input signal for target peripheral
trigsel_register_lock_set	lock the trigger register
trigsel_register_lock_get	get the trigger register lock status

Enum trigsel_source_enum

Table 3-762. Enum trigsel_source_enum

Member name	Function description
TRIGSEL_INPUT_0	trigger input source 0
TRIGSEL_INPUT_1	trigger input source 1
TRIGSEL_INPUT_TRIGSEL_IN0	trigger input source TRIGSEL_IN0 pin
TRIGSEL_INPUT_TRIGSEL_IN1	trigger input source TRIGSEL_IN1 pin
TRIGSEL_INPUT_TRIGSEL_IN2	trigger input source TRIGSEL_IN2 pin
TRIGSEL_INPUT_TRIGSEL_IN3	trigger input source TRIGSEL_IN3 pin
TRIGSEL_INPUT_TRIGSEL_IN4	trigger input source TRIGSEL_IN4 pin
TRIGSEL_INPUT_TRIGSEL_IN5	trigger input source TRIGSEL_IN5 pin
TRIGSEL_INPUT_TRIGSEL_IN6	trigger input source TRIGSEL_IN6 pin
TRIGSEL_INPUT_TRIGSEL_IN7	trigger input source TRIGSEL_IN7 pin
TRIGSEL_INPUT_TRIGSEL_IN8	trigger input source TRIGSEL_IN8 pin
TRIGSEL_INPUT_TRIGSEL_IN9	trigger input source TRIGSEL_IN9 pin
TRIGSEL_INPUT_TRIGSEL_IN10	trigger input source TRIGSEL_IN10 pin
TRIGSEL_INPUT_TRIGSEL_IN11	trigger input source TRIGSEL_IN11 pin
TRIGSEL_INPUT_CMP_OUT	trigger input source CMP_OUT
TRIGSEL_INPUT_LXTAL_TRG	trigger input source LXTAL_TRG
TRIGSEL_INPUT_TIMER1_CH0	trigger input source timer1 channel 0
TRIGSEL_INPUT_TIMER1_CH1	trigger input source timer1 channel 1
TRIGSEL_INPUT_TIMER1_CH2	trigger input source timer1 channel 2
TRIGSEL_INPUT_TIMER1_CH3	trigger input source timer1 channel 3
TRIGSEL_INPUT_TIMER1_TRGO	trigger input source timer1 TRGO
TRIGSEL_INPUT_TIMER0_CH0	trigger input source timer0 channel 0
TRIGSEL_INPUT_TIMER0_CH1	trigger input source timer0 channel 1
TRIGSEL_INPUT_TIMER0_CH2	trigger input source timer0 channel 2
TRIGSEL_INPUT_TIMER0_CH3	trigger input source timer0 channel 3
TRIGSEL_INPUT_TIMER0_MCH0	trigger input source timer0 multi mode channel 0
TRIGSEL_INPUT_TIMER0_MCH1	trigger input source timer0 multi mode channel 1
TRIGSEL_INPUT_TIMER0_MCH2	trigger input source timer0 multi mode channel 2

Member name	Function description
TRIGSEL_INPUT_TIMER0_MCH3	trigger input source timer0 multi mode channel 3
TRIGSEL_INPUT_TIMER0_TRGO	trigger input source timer0 TRGO
TRIGSEL_INPUT_TIMER7_CH0	trigger input source timer7 channel 0
TRIGSEL_INPUT_TIMER7_CH1	trigger input source timer7 channel 1
TRIGSEL_INPUT_TIMER7_CH2	trigger input source timer7 channel 2
TRIGSEL_INPUT_TIMER7_CH3	trigger input source timer7 channel 3
TRIGSEL_INPUT_TIMER7_MCH0	trigger input source timer7 multi mode channel 0
TRIGSEL_INPUT_TIMER7_MCH1	trigger input source timer7 multi mode channel 1
TRIGSEL_INPUT_TIMER7_MCH2	trigger input source timer7 multi mode channel 2
TRIGSEL_INPUT_TIMER7_MCH3	trigger input source timer7 multi mode channel 3
TRIGSEL_INPUT_TIMER7_TRGO	trigger input source timer7 TRGO
TRIGSEL_INPUT_TIMER19_CH0	trigger input source timer19 channel 0
TRIGSEL_INPUT_TIMER19_CH1	trigger input source timer19 channel 1
TRIGSEL_INPUT_TIMER19_CH2	trigger input source timer19 channel 2
TRIGSEL_INPUT_TIMER19_CH3	trigger input source timer19 channel 3
TRIGSEL_INPUT_TIMER19_MCH0	trigger input source timer19 multi mode channel 0
TRIGSEL_INPUT_TIMER19_MCH1	trigger input source timer19 multi mode channel 1
TRIGSEL_INPUT_TIMER19_MCH2	trigger input source timer19 multi mode channel 2
TRIGSEL_INPUT_TIMER19_MCH3	trigger input source timer19 multi mode channel 3
TRIGSEL_INPUT_TIMER19_TRGO	trigger input source timer19 TRGO
TRIGSEL_INPUT_TIMER20_CH0	trigger input source timer20 channel 0
TRIGSEL_INPUT_TIMER20_CH1	trigger input source timer20 channel 1
TRIGSEL_INPUT_TIMER20_CH2	trigger input source timer20 channel 2
TRIGSEL_INPUT_TIMER20_CH3	trigger input source timer20 channel 3
TRIGSEL_INPUT_TIMER20_MCH0	trigger input source timer20 multi mode channel 0
TRIGSEL_INPUT_TIMER20_MCH1	trigger input source timer20 multi mode channel 1
TRIGSEL_INPUT_TIMER20_MCH2	trigger input source timer20 multi mode channel 2
TRIGSEL_INPUT_TIMER20_MCH3	trigger input source timer20 multi mode channel 3
TRIGSEL_INPUT_TIMER20_TRGO	trigger input source timer20 TRGO
TRIGSEL_INPUT_TIMER5_TRGO	trigger input source timer5 TRGO
TRIGSEL_INPUT_TIMER6_TRGO	trigger input source timer6 TRGO
TRIGSEL_INPUT_MFCOM_TRIG0	trigger input source MFCOM TRIG0
TRIGSEL_INPUT_MFCOM_TRIG1	trigger input source MFCOM TRIG1
TRIGSEL_INPUT_MFCOM_TRIG2	trigger input source MFCOM TRIG2
TRIGSEL_INPUT_MFCOM_TRIG3	trigger input source MFCOM TRIG3
TRIGSEL_INPUT_RTC_ALARM	trigger input source RTC alarm
TRIGSEL_INPUT_RTC_SECOND	trigger input source RTC second
TRIGSEL_INPUT_TRIGSEL_IN12	trigger input source TRIGSEL_IN12 pin
TRIGSEL_INPUT_TRIGSEL_IN13	trigger input source TRIGSEL_IN13 pin

Enum trigsel_periph_enum

Table 3-763. Enum trigsel_periph_enum

Member name	Function description
TRIGSEL_OUTPUT_TRIGSEL_OUT0	output target peripheral TRIGSEL_OUT0 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT1	output target peripheral TRIGSEL_OUT1 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT2	output target peripheral TRIGSEL_OUT2 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT3	output target peripheral TRIGSEL_OUT3 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT4	output target peripheral TRIGSEL_OUT4 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT5	output target peripheral TRIGSEL_OUT5 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT6	output target peripheral TRIGSEL_OUT6 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT7	output target peripheral TRIGSEL_OUT7 pin
TRIGSEL_OUTPUT_ADC0_RTTRG	output target peripheral ADC0_RTTRG
TRIGSEL_OUTPUT_ADC0_INSTRG	output target peripheral ADC0_INSTRG
TRIGSEL_OUTPUT_ADC1_RTTRG	output target peripheral ADC1_RTTRG
TRIGSEL_OUTPUT_ADC1_INSTRG	output target peripheral ADC1_INSTRG
TRIGSEL_OUTPUT_DAC_EXTRIG	output target peripheral DAC_EXTRIG
TRIGSEL_OUTPUT_TIMER0_ITI0	output target peripheral TIMER0_ITI0
TRIGSEL_OUTPUT_TIMER0_ITI1	output target peripheral TIMER0_ITI1
TRIGSEL_OUTPUT_TIMER0_ITI2	output target peripheral TIMER0_ITI2
TRIGSEL_OUTPUT_TIMER0_ITI3	output target peripheral TIMER0_ITI3
TRIGSEL_OUTPUT_TIMER0_BRKIN0	output target peripheral TIMER0_BRKIN0
TRIGSEL_OUTPUT_TIMER0_BRKIN1	output target peripheral TIMER0_BRKIN1
TRIGSEL_OUTPUT_TIMER0_BRKIN2	output target peripheral TIMER0_BRKIN2
TRIGSEL_OUTPUT_TIMER0_BRKIN3	output target peripheral TIMER0_BRKIN3
TRIGSEL_OUTPUT_TIMER7_ITI0	output target peripheral TIMER7_ITI0
TRIGSEL_OUTPUT_TIMER7_ITI1	output target peripheral TIMER7_ITI1
TRIGSEL_OUTPUT_TIMER7_ITI2	output target peripheral TIMER7_ITI2
TRIGSEL_OUTPUT_TIMER7_ITI3	output target peripheral TIMER7_ITI3
TRIGSEL_OUTPUT_TIMER7_BRKIN0	output target peripheral TIMER7_BRKIN0
TRIGSEL_OUTPUT_TIMER7_BRKIN1	output target peripheral TIMER7_BRKIN1
TRIGSEL_OUTPUT_TIMER7_BRKIN2	output target peripheral TIMER7_BRKIN2
TRIGSEL_OUTPUT_TIMER7_BRKIN3	output target peripheral TIMER7_BRKIN3
TRIGSEL_OUTPUT_TIMER19_ITI0	output target peripheral TIMER19_ITI0
TRIGSEL_OUTPUT_TIMER19_ITI1	output target peripheral TIMER19_ITI1
TRIGSEL_OUTPUT_TIMER19_ITI2	output target peripheral TIMER19_ITI2
TRIGSEL_OUTPUT_TIMER19_ITI3	output target peripheral TIMER19_ITI3
TRIGSEL_OUTPUT_TIMER19_BRKIN0	output target peripheral TIMER19_BRKIN0
TRIGSEL_OUTPUT_TIMER19_BRKIN1	output target peripheral TIMER19_BRKIN1
TRIGSEL_OUTPUT_TIMER19_BRKIN2	output target peripheral TIMER19_BRKIN2
TRIGSEL_OUTPUT_TIMER19_BRKIN3	output target peripheral TIMER19_BRKIN3
TRIGSEL_OUTPUT_TIMER20_ITI0	output target peripheral TIMER20_ITI0

Member name	Function description
TRIGSEL_OUTPUT_TIMER20_ITI1	output target peripheral TIMER20_ITI1
TRIGSEL_OUTPUT_TIMER20_ITI2	output target peripheral TIMER20_ITI2
TRIGSEL_OUTPUT_TIMER20_ITI3	output target peripheral TIMER20_ITI3
TRIGSEL_OUTPUT_TIMER20_BRKIN0	output target peripheral TIMER20_BRKIN0
TRIGSEL_OUTPUT_TIMER20_BRKIN1	output target peripheral TIMER20_BRKIN1
TRIGSEL_OUTPUT_TIMER20_BRKIN2	output target peripheral TIMER20_BRKIN2
TRIGSEL_OUTPUT_TIMER20_BRKIN3	output target peripheral TIMER20_BRKIN3
TRIGSEL_OUTPUT_TIMER1_ITI0	output target peripheral TIMER1_ITI0
TRIGSEL_OUTPUT_TIMER1_ITI1	output target peripheral TIMER1_ITI1
TRIGSEL_OUTPUT_TIMER1_ITI2	output target peripheral TIMER1_ITI2
TRIGSEL_OUTPUT_TIMER1_ITI3	output target peripheral TIMER1_ITI3
TRIGSEL_OUTPUT_MFCOM_TRG_TIME R0	output target peripheral MFCOM_TRG_TIMER0
TRIGSEL_OUTPUT_MFCOM_TRG_TIME R1	output target peripheral MFCOM_TRG_TIMER1
TRIGSEL_OUTPUT_MFCOM_TRG_TIME R2	output target peripheral MFCOM_TRG_TIMER2
TRIGSEL_OUTPUT_MFCOM_TRG_TIME R3	output target peripheral MFCOM_TRG_TIMER3
TRIGSEL_OUTPUT_CAN0_EX_TIME_TIC K	output target peripheral CAN0_EX_TIME_TICK
TRIGSEL_OUTPUT_CAN1_EX_TIME_TIC K	output target peripheral CAN1_EX_TIME_TICK

trigsel_init

The description of trigsel_init is shown as below:

Table 3-764. Function trigsel_init

Function name	trigsel_init
Function prototype	void trigsel_init(trigsel_periph_enum target_periph, trigsel_source_enum trigger_source);
Function descriptions	set the trigger input signal for target peripheral
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	target peripheral value, refer to Table 3-763. Enum trigsel_periph_enum
Input parameter{in}	
trigger_source	trigger source value, refer to Table 3-762. Enum trigsel_source_enum
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* select TIMER0_CH2 to trigger ADC0 */
trigsel_init(TRIGSEL_OUTPUT_ADC0_RTTRG, TRIGSEL_INPUT_TIMER0_CH2);
```

trigsel_trigger_source_get

The description of trigsel_trigger_source_get is shown as below:

Table 3-765. Function trigsel_trigger_source_get

Function name	trigsel_trigger_source_get
Function prototype	uint8_t trigsel_trigger_source_get(trigsel_periph_enum target_periph);
Function descriptions	get the trigger input signal for target peripheral
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	target peripheral value, refer to Table 3-763. Enum trigsel_periph_enum
Output parameter{out}	
-	-
Return value	
trigger_source	trigger source value, the value scope should be 0-67

Example:

```
/* get the trigger input signal for ADC0 */
uint8_t input_signal;
input_signal = trigsel_trigger_source_get(TRIGSEL_OUTPUT_ADC0_RTTRG);
```

trigsel_register_lock_set

The description of trigsel_register_lock_set is shown as below:

Table 3-766. Function trigsel_register_lock_set

Function name	trigsel_register_lock_set
Function prototype	void trigsel_register_lock_set(trigsel_periph_enum target_periph);
Function descriptions	lock the trigger register
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	target peripheral value, refer to Table 3-763. Enum trigsel_periph_enum
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* lock the trigger register for ADC0 */
trigsel_register_lock_set(TRIGSEL_OUTPUT_ADC0_RTTRG);
```

trigsel_register_lock_get

The description of trigsel_register_lock_get is shown as below:

Table 3-767. Function trigsel_register_lock_get

Function name	trigsel_register_lock_get
Function prototype	FlagStatus trigsel_register_lock_get(trigsel_periph_enum target_periph);
Function descriptions	get the trigger register lock status
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	target peripheral value, refer to Table 3-763. Enum trigsel_periph_enum
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the trigger register lock status of ADC0 */
FlagStatus status;
status = trigsel_register_lock_get(TRIGSEL_OUTPUT_ADC0_RTTRG);
```

3.24. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter [3.24.1](#), the USART firmware functions are introduced in chapter [3.24.2](#).

3.24.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

Table 3-768. USART Registers

Registers	Descriptions
USART_CTL0	USART control register 0
USART_CTL1	USART control register 1

Registers	Descriptions
USART_CTL2	USART control register 2
USART_BAUD	USART baud rate register
USART_GP	USART guard time and prescaler register
USART_RT	USART receiver timeout register
USART_CMD	USART command register
USART_STAT	USART status register
USART_INTC	USART status clear register
USART_RDATA	USART receive data register
USART_TDATA	USART transmit data register
USART_CHC	USART coherence control register
USART_RFCS	USART receive FIFO control and status register

3.24.2. Descriptions of Peripheral functions

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

Table 3-769. 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_command_enable	enable USART command
usart_address_config	configure address of the USART
usart_address_detection_mode_config	configure address detection mode

Function name	Function description
g	
usart_mute_mode_enable	enable mute mode
usart_mute_mode_disable	disable mute mode
usart_mute_mode_wakeup_config	configure wakeup method in mute mode
usart_lin_mode_enable	enable LIN mode
usart_lin_mode_disable	disable LIN mode
usart_lin_break_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

Function name	Function description
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
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_flag_get	get USART status
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 3-770. 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 3-771. 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 3-772. 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 3-773. 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 3-774. 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 3-775. 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 3-776. 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 USART0 parity */
usart_parity_config(USART0, USART_PM_EVEN);
```


usart_word_length_set

The description of usart_word_length_set is shown as below:

Table 3-777. 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
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
wlen	USART word length configure
<i>USART_WL_8BIT</i>	8 bits
<i>USART_WL_9BIT</i>	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 3-778. 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
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
stblen	USART stop bit configure
<i>USART_STB_1BIT</i>	1 bit
<i>USART_STB_0_5BIT</i>	0.5 bit
<i>USART_STB_2BIT</i>	2 bits

USART_STB_1_5BIT	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 3-779. 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
USARTx	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 3-780. 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>	x=0,1,2
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 3-781. 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>	x=0,1,2
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 3-782. Function `usart_receive_config`

Function name	<code>usart_receive_config</code>
Function prototype	<code>void usart_receive_config(uint32_t usart_periph, uint32_t rxconfig);</code>
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 3-783. Function `usart_data_first_config`

Function name	<code>usart_data_first_config</code>
Function prototype	<code>void usart_data_first_config(uint32_t usart_periph, uint32_t msbf);</code>
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 LSB of data first */
```

```
usart_data_first_config(USART0, USART_MSBF_LSB);
```

usart_invert_config

The description of usart_invert_config is shown as below:

Table 3-784. Function usart_invert_config

Function name	usart_invert_config
Function prototype	void usart_invert_config(uint32_t usart_periph, usart_invert_enum invertpara);
Function descriptions	configure USART inverted
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
invertpara	refer to Table 3-773. Enum usart_invert_enum
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 inversion */
```

```
usart_invert_config(USART0, USART_DINV_ENABLE);
```

usart_overrun_enable

The description of usart_overrun_enable is shown as below:

Table 3-785. Function usart_overrun_enable

Function name	usart_overrun_enable
Function prototype	void usart_overrun_enable(uint32_t usart_periph);
Function descriptions	enable the USART overrun 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 USART0 overrun */
```

```
usart_overrun_enable(USART0);
```

usart_overrun_disable

The description of usart_overrun_disable is shown as below:

Table 3-786. Function usart_overrun_disable

Function name	usart_overrun_disable
Function prototype	void usart_overrun_disable(uint32_t usart_periph);
Function descriptions	disable the USART overrun 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 USART0 overrun */
```

```
usart_overrun_disable(USART0);
```

usart_oversample_config

The description of usart_oversample_config is shown as below:

Table 3-787. 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>	oversampling by 8
<i>USART_OVSMOD_16</i>	oversampling by 16
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config USART0 oversampling by 8 */
```

```
usart_oversample_config(USART0, USART_OVSMOD_8);
```

usart_sample_bit_config

The description of usart_sample_bit_config is shown as below:

Table 3-788. 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 bit
<i>USART_OSB_3BIT</i>	3 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config USART0 1 bit sample mode */
```

```
usart_sample_bit_config(USART0, USART_OSB_1BIT);
```

usart_receiver_timeout_enable

The description of usart_receiver_timeout_enable is shown as below:

Table 3-789. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 receiver timeout */
usart_receiver_timeout_enable(USART0);
```

usart_receiver_timeout_disable

The description of usart_receiver_timeout_disable is shown as below:

Table 3-790. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 receiver timeout */
usart_receiver_timeout_disable(USART0);
```


usart_receiver_timeout_threshold_config

The description of usart_receiver_timeout_threshold_config is shown as below:

Table 3-791. Function usart_receiver_timeout_threshold_config

Function name	usart_receiver_timeout_threshold_config
Function prototype	void usart_receiver_timeout_threshold_config(uint32_t usart_periph, uint32_t rtimeout);
Function descriptions	configure receiver timeout threshold
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Input parameter{in}	
rtimeout	receiver timeout (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 3-792. Function usart_data_transmit

Function name	usart_data_transmit
Function prototype	void usart_data_transmit(uint32_t usart_periph, uint16_t data);
Function descriptions	USART transmit data function
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
data	data of transmission (0x0000-0x01FF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 transmit data */

usart_data_transmit(USART0, 0x00AA);
```

usart_data_receive

The description of usart_data_receive is shown as below:

Table 3-793. 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	
uint16_t	data of received (0x0000-0x01FF)

Example:

```
/* USART0 receive data */

uint16_t temp;

temp = usart_data_receive(USART0);
```

usart_command_enable

The description of usart_command_enable is shown as below:

Table 3-794. 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	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
cmdtype	command type

<i>USART_CMD_SBKCM</i> <i>D</i>	send break command
<i>USART_CMD_MMCM</i> <i>D</i>	mute mode command
<i>USART_CMD_RXFCM</i> <i>D</i>	receive data flush command
<i>USART_CMD_TXFCM</i> <i>D</i>	transmit data flush request
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 command */
```

```
usart_command_enable(USART0, USART_CMD_SBKCMD);
```

usart_address_config

The description of usart_address_config is shown as below:

Table 3-795. Function usart_address_config

Function name	usart_address_config
Function prototype	void usart_address_config(uint32_t usart_periph, uint8_t addr);
Function descriptions	configure the address of the USART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
addr	address of USART (0x00-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address of the USART0 */
```

```
usart_address_config(USART0, 0x00);
```

usart_address_detection_mode_config

The description of usart_address_detection_mode_config is shown as below:

Table 3-796. 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}	
usart_periph	USARTx peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
addmod	address detection mode
<i>USART_ADDDM_4BIT</i>	4 bits
<i>USART_ADDDM_FULLBIT</i>	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 3-797. Function `usart_mute_mode_enable`

Function name	<code>usart_mute_mode_enable</code>
Function prototype	<code>void usart_mute_mode_enable(uint32_t usart_periph);</code>
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 3-798. 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 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 3-799. 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 mask

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 wakeup method in mute mode */
usart_mute_mode_wakeup_config(USART0, USART_WM_IDLE);
```

usart_lin_mode_enable

The description of usart_lin_mode_enable is shown as below:

Table 3-800. 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	usart peripheral
USARTx	x=0,1,2
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 3-801. 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	usart peripheral

<i>USARTx</i>	<i>x</i> =0,1,2
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 3-802. Function usart_lin_break_dection_length_config

Function name	usart_lin_break_dection_length_config
Function prototype	void usart_lin_break_dection_length_config(uint32_t usart_periph, uint32_t lblen);
Function descriptions	LIN break detection length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	<i>x</i> =0,1,2
Input parameter{in}	
lblen	two methods be used to enter or exit the mute mode
<i>USART_LBLEN_10B</i>	10 bits
<i>USART_LBLEN_11B</i>	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 3-803. 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 3-804. 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 3-805. Function usart_clock_enable

Function name	usart_clock_enable
Function prototype	void usart_clock_enable(uint32_t usart_periph);
Function descriptions	enable clock
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable clock */
usart_clock_enable(USART0);
```

usart_clock_disable

The description of usart_clock_disable is shown as below:

Table 3-806. Function usart_clock_disable

Function name	usart_clock_disable
Function prototype	void usart_clock_disable(uint32_t usart_periph);
Function descriptions	disable clock
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable clock */
usart_clock_disable(USART0);
```

usart_synchronous_clock_config

The description of usart_synchronous_clock_config is shown as below:

Table 3-807. Function `usart_synchronous_clock_config`

Function name	<code>usart_synchronous_clock_config</code>
Function prototype	<code>void usart_synchronous_clock_config(uint32_t usart_periph, uint32_t clen, uint32_t cph, uint32_t cpl);</code>
Function descriptions	configure USART synchronous mode parameters
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
clen	last bit clock pulse
<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 3-808. Function `usart_guard_time_config`

Function name	<code>usart_guard_time_config</code>
Function prototype	<code>void usart_guard_time_config(uint32_t usart_periph,uint32_t guat);</code>
Function descriptions	configure guard time value in smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	

usart_periph	usart peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
guat	guard time value (0x00000000-0x000000FF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 guard time value in smartcard mode */
```

```
usart_guard_time_config(USART0, 0x00000055);
```

usart_smartcard_mode_enable

The description of usart_smartcard_mode_enable is shown as below:

Table 3-809. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 smartcard mode enable */
```

```
usart_smartcard_mode_enable(USART0);
```

usart_smartcard_mode_disable

The description of usart_smartcard_mode_disable is shown as below:

Table 3-810. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* USART0 smartcard mode disable */
usart_smartcard_mode_disable(USART0);
```

usart_smartcard_mode_nack_enable

The description of usart_smartcard_mode_nack_enable is shown as below:

Table 3-811. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 NACK in smartcard mode */
usart_smartcard_mode_nack_enable(USART0);
```

usart_smartcard_mode_nack_disable

The description of usart_smartcard_mode_nack_disable is shown as below:

Table 3-812. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
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 3-813. 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>	x=0,1,2
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 3-814. 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>	x=0,1,2
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 3-815. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
scrtnum	smartcard auto-retry number (0x00000000-0x00000007)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure smartcard auto-retry number */
```

```
usart_smartcard_autoretry_config(USART0, 0x00000007);
```

usart_block_length_config

The description of usart_block_length_config is shown as below:

Table 3-816. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
bl	block length(0x00000000-0x000000FF)
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 3-817. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
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 3-818. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
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 3-819. Function usart_prescaler_config

Function name	usart_prescaler_config
Function prototype	void usart_prescaler_config(uint32_t usart_periph, uint32_t psc);
Function descriptions	configure the peripheral clock prescaler in USART IrDA low-power or smartcard mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
psc	clock prescaler (0x00000000-0x000000FF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the USART0 peripheral clock prescaler in USART IrDA low-power mode */
```

```
usart_prescaler_config(USART0, 0x00000001);
```

usart_irda_lowpower_config

The description of usart_irda_lowpower_config is shown as below:

Table 3-820. 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	usart peripheral
USARTx	x=0,1,2
Input parameter{in}	
irlp	IrDA low-power or normal
USART_IRLP_LOW	low-power
USART_IRLP_NORMAL	normal
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 IrDA low-power */
```

```
usart_irda_lowpower_config(USART0, USART_IRLP_LOW);
```

usart_hardware_flow_rts_config

The description of usart_hardware_flow_rts_config is shown as below:

Table 3-821. 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	usart 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 3-822. 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 CTS
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart 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 3-823. 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	usart peripheral
USARTx	x=0,1,2
Input parameter{in}	
hcm	hardware flow control coherence mode
USART_HCM_NONE	nRTS signal equals to the rxne status register
USART_HCM_EN	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 3-824. Function usart_rs485_driver_enable

Function name	usart_rs485_driver_enable
Function prototype	void usart_rs485_driver_enable(uint32_t usart_periph);
Function descriptions	enable USART RS485 driver
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	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 3-825. Function usart_rs485_driver_disable

Function name	usart_rs485_driver_disable
Function prototype	void usart_rs485_driver_disable(uint32_t usart_periph);
Function descriptions	disable USART RS485 driver
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart 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_asserttime_config

The description of usart_driver_asserttime_config is shown as below:

Table 3-826. Function usart_driver_asserttime_config

Function name	usart_driver_asserttime_config
Function prototype	void usart_driver_asserttime_config(uint32_t usart_periph, uint32_t deatime);
Function descriptions	configure driver enable assertion time
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
deatime	driver enable assertion time (0x00000000-0x0000001F)
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* set USART0 driver assertime */
```

```
usart_driver_assertime_config(USART0, 0x0000001F);
```

usart_driver_deassertime_config

The description of usart_driver_deassertime_config is shown as below:

Table 3-827. Function usart_driver_deassertime_config

Function name	usart_driver_deassertime_config
Function prototype	void usart_driver_deassertime_config(uint32_t usart_periph, uint32_t dedtime);
Function descriptions	configure driver enable de-assertion time
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
dedtime	driver enable de-assertion time (0x00000000-0x0000001F)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set USART0 driver deassertime */
```

```
usart_driver_deassertime_config(USART0, 0x0000001F);
```

usart_depolarity_config

The description of usart_depolarity_config is shown as below:

Table 3-828. 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	usart peripheral
<i>USARTx</i>	x=0,1,2
Input parameter{in}	
dep	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_depolarity_config(USART0, USART_DEP_HIGH);
```

usart_dma_receive_config

The description of usart_dma_receive_config is shown as below:

Table 3-829. Function usart_dma_receive_config

Function name	usart_dma_receive_config
Function prototype	void usart_dma_receove_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:

```
/* enable USART0 DMA for reception */
```

```
usart_dma_receive_config(USART0, USART_RECEIVE_DMA_ENABLE);
```

usart_dma_transmit_config

The description of usart_dma_transmit_config is shown as below:

Table 3-830. 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:

```
/* enable USART0 DMA for 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 3-831. 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 3-832. 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 3-833. 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=0,1,2
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 3-834. 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
USARTx	x=0,1,2
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 3-835. 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=0,1,2
Input parameter{in}	

wum	wakeup mode
<i>USART_WUM_ADDR</i>	WUF active on address match
<i>USART_WUM_START</i> <i>B</i>	WUF active on start bit
<i>USART_WUM_RBNE</i>	WUF active on RBNE
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 wake up mode */
usart_wakeup_mode_config(USART0, USART_WUM_ADDR);
```

usart_receive_fifo_enable

The description of usart_receive_fifo_enable is shown as below:

Table 3-836. 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	USARTx peripheral
<i>USARTx</i>	x=0,1,2
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 3-837. 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	USARTx peripheral
<i>USARTx</i>	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 3-838. 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	USARTx peripheral
<i>USARTx</i>	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_flag_get

The description of usart_flag_get is shown as below:

Table 3-839. Function `usart_flag_get`

Function name	<code>usart_flag_get</code>
Function prototype	<code>FlagStatus usart_flag_get(uint32_t usart_periph, usart_flag_enum flag);</code>
Function descriptions	get flag in STAT/RFCs register
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	USARTx peripheral
<code>USARTx</code>	x=0,1,2
Input parameter{in}	
<code>flag</code>	USART flags, refer to Table 3-770. Enum usart_flag_enum only one among these parameters can be selected
Output parameter{out}	
-	-
Return value	
<code>FlagStatus</code>	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 3-840. Function `usart_flag_clear`

Function name	<code>usart_flag_clear</code>
Function prototype	<code>void usart_flag_clear(uint32_t usart_periph, usart_flag_enum flag);</code>
Function descriptions	clear flag in STAT register
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	USARTx peripheral
<code>USARTx</code>	x=0,1,2
Input parameter{in}	
<code>flag</code>	USART flags, refer to Table 3-770. Enum usart_flag_enum only one among these parameters can be selected
<code>USART_FLAG_WU</code>	wakeup from deep-sleep mode flag
<code>USART_FLAG_AM</code>	ADDR match flag
<code>USART_FLAG_EB</code>	end of block flag
<code>USART_FLAG_RT</code>	receiver timeout flag

USART_FLAG_CTSF	CTS change flag
USART_FLAG_LBD	LIN break detected flag
USART_FLAG_TC	transmission complete
USART_FLAG_RBNE	read data buffer not empty
USART_FLAG_IDLE	IDLE line detected flag
USART_FLAG_ORER R	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
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 3-841. 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	interrupt type, refer to Table 3-772. Enum usart_interrupt_enum only one among these parameters can be selected
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 3-842. 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}	
interrupt	interrupt type, refer to Table 3-772. Enum usart_interrupt_enum only one among these parameters can be selected
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 3-843. 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 and flag status
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 3-771. Enum usart_interrupt_flag_enum , only one among these parameters can be selected
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 3-844. 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
USARTx	x=0,1,2
Input parameter{in}	
int_flag	USART interrupt flag, refer to Table 3-771. Enum usart_interrupt_flag_enum , only one among these parameters can be selected
USART_INT_FLAG_EB	end of block interrupt and flag
USART_INT_FLAG_RT	receiver timeout interrupt and flag
USART_INT_FLAG_A M	address match interrupt and flag
USART_INT_FLAG_PE RR	parity error interrupt and flag
USART_INT_FLAG_TC	transmission complete interrupt and flag
USART_INT_FLAG_RB NE_ORERR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_ID LE	IDLE line detected interrupt and flag

USART_INT_FLAG_LB D	LIN break detected interrupt and flag
USART_INT_FLAG_W U	wakeup from deep-sleep mode interrupt and flag
USART_INT_FLAG_CT S	CTS interrupt and flag
USART_INT_FLAG_ER R_NERR	error interrupt and noise error flag
USART_INT_FLAG_ER R_ORERR	error interrupt and overrun error
USART_INT_FLAG_ER R_FERR	error interrupt and frame error flag
USART_INT_FLAG_RF F	receive FIFO full interrupt and flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the USART0 interrupt flag */
```

```
usart_interrupt_flag_clear(USART0, USART_INT_FLAG_TC);
```

3.25. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. The WWDGT registers are listed in chapter [3.25.1](#), the WWDGT firmware functions are introduced in chapter [3.25.2](#).

3.25.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

Table 3-845. WWDGT Registers

Registers	Descriptions
WWDGT_CTL	WWDGT control register
WWDGT_CFG	WWDGT configuration register
WWDGT_STAT	WWDGT status register

3.25.2. Descriptions of Peripheral functions

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

Table 3-846. WWDGT firmware function

Function name	Function description
wwdgt_deinit	reset the window watchdog timer configuration
wwdgt_enable	start the window watchdog timer counter
wwdgt_counter_update	configure the window watchdog timer counter value
wwdgt_config	configure counter value, window value, and prescaler divider value
wwdgt_interrupt_enable	enable early wakeup interrupt of WWDGT
wwdgt_flag_get	check early wakeup interrupt state of WWDGT
wwdgt_flag_clear	clear early wakeup interrupt state of WWDGT

wwdgt_deinit

The description of wwdgt_deinit is shown as below:

Table 3-847. Function wwdgt_deinit

Function name	wwdgt_deinit
Function prototype	void wwdgt_deinit(void);
Function descriptions	reset the window watchdog timer configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the window watchdog timer configuration */
```

```
wwdgt_deinit();
```

wwdgt_enable

The description of wwdgt_enable is shown as below:

Table 3-848. Function wwdgt_enable

Function name	wwdgt_enable
Function prototype	void wwdgt_enable(void);
Function descriptions	start the window watchdog timer counter
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start the WWDGT counter */
wwdgt_enable();
```

wwdgt_counter_update

The description of wwdgt_counter_update is shown as below:

Table 3-849. Function wwdgt_counter_update

Function name	wwdgt_counter_update
Function prototype	void wwdgt_counter_update(uint16_t counter_value);
Function descriptions	configure the window watchdog timer counter value
Precondition	-
The called functions	-
Input parameter{in}	
counter_value	counter_value: 0x00000000 - 0x0000007F
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* update WWDGT counter to 0x7F */
wwdgt_counter_update(127);
```

wwdgt_config

The description of wwdgt_config is shown as below:

Table 3-850. 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	counter: 0x00000000 - 0x0000007F
Input parameter{in}	

window	window: 0x00000000 - 0x0000007F
Input parameter{in}	
prescaler	wwdgt prescaler value
WWDGT_CFG_PSC_D IV1	the time base of WWDGT counter = (PCLK1/4096)/1
WWDGT_CFG_PSC_D IV2	the time base of WWDGT counter = (PCLK1/4096)/2
WWDGT_CFG_PSC_D IV4	the time base of WWDGT counter = (PCLK1/4096)/4
WWDGT_CFG_PSC_D IV8	the time base of WWDGT counter = (PCLK1/4096)/8
WWDGT_CFG_PSC_D IV16	the time base of WWDGT counter = (PCLK1/4096)/16
WWDGT_CFG_PSC_D IV32	the time base of WWDGT counter = (PCLK1/4096)/32
WWDGT_CFG_PSC_D IV64	the time base of WWDGT counter = (PCLK1/4096)/64
WWDGT_CFG_PSC_D IV128	the time base of WWDGT counter = (PCLK1/4096)/128
WWDGT_CFG_PSC_D IV256	the time base of WWDGT counter = (PCLK1/4096)/256
WWDGT_CFG_PSC_D IV512	the time base of WWDGT counter = (PCLK1/4096)/512
WWDGT_CFG_PSC_D IV1024	the time base of WWDGT counter = (PCLK1/4096)/1024
WWDGT_CFG_PSC_D IV2048	the time base of WWDGT counter = (PCLK1/4096)/2048
WWDGT_CFG_PSC_D IV4096	the time base of WWDGT counter = (PCLK1/4096)/4096
WWDGT_CFG_PSC_D IV8192	the time base of WWDGT counter = (PCLK1/4096)/8192
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure WWDGT counter value to 0x7F, window value to 0x50, prescaler divider value to 8 */
```

```
wwdgt_config(127, 80, WWDGT_CFG_PSC_DIV8);
```

wwdgt_interrupt_enable

The description of wwdgt_interrupt_enable is shown as below:

Table 3-851. 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 3-852. 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 3-853. 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	Aug. 20, 2023

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.