

**GigaDevice Semiconductor Inc.**

**GD32G5x3**

**Arm<sup>®</sup> Cortex<sup>®</sup>-M33 32-bit MCU**

**Firmware Library  
User Guide**

Revision 1.1

(Feb. 2025)

# Table of Contents

<b>Table of Contents .....</b>	<b>1</b>
<b>List of Figures .....</b>	<b>5</b>
<b>List of Tables .....</b>	<b>6</b>
<b>1. Introduction .....</b>	<b>39</b>
<b>1.1. Rules of User Manual and Firmware Library .....</b>	<b>39</b>
1.1.1. Peripherals.....	39
1.1.2. Naming rules.....	40
<b>2. Firmware Library Overview .....</b>	<b>42</b>
<b>2.1. File Structure of Firmware Library .....</b>	<b>42</b>
2.1.1. Examples Folder .....	43
2.1.2. Firmware Folder .....	43
2.1.3. Template Folder .....	43
2.1.4. Utilities Folder .....	45
<b>2.2. File descriptions of Firmware Library .....</b>	<b>46</b>
<b>3. Firmware Library of Standard Peripherals .....</b>	<b>47</b>
<b>3.1. Overview of Firmware Library of Standard Peripherals.....</b>	<b>47</b>
<b>3.2. ADC .....</b>	<b>47</b>
3.2.1. Descriptions of Peripheral registers.....	47
3.2.2. Descriptions of Peripheral functions .....	48
<b>3.3. CAN .....</b>	<b>90</b>
3.3.1. Descriptions of Peripheral registers.....	90
3.3.2. Descriptions of Peripheral functions .....	92
<b>3.4. CAU .....</b>	<b>137</b>
3.4.1. Descriptions of Peripheral registers.....	137
3.4.2. Descriptions of Peripheral functions .....	137
<b>3.5. CMP .....</b>	<b>165</b>
3.5.1. Descriptions of Peripheral registers.....	165
3.5.2. Descriptions of Peripheral functions .....	165
<b>3.6. CPDM.....</b>	<b>179</b>
3.6.1. Descriptions of Peripheral registers.....	180
3.6.2. Descriptions of Peripheral functions .....	180
<b>3.7. CRC .....</b>	<b>186</b>
3.7.1. Descriptions of Peripheral registers.....	186

3.7.2.	Descriptions of Peripheral functions .....	186
<b>3.8.</b>	<b>DAC .....</b>	<b>194</b>
3.8.1.	Description of peripheral registers .....	194
3.8.2.	Descriptions of Peripheral functions .....	195
<b>3.9.</b>	<b>DBG .....</b>	<b>221</b>
3.9.1.	Descriptions of Peripheral registers .....	221
3.9.2.	Descriptions of Peripheral functions .....	222
<b>3.10.</b>	<b>DMA / DMAMUX .....</b>	<b>227</b>
3.10.1.	Descriptions of Peripheral registers .....	227
3.10.2.	Descriptions of Peripheral functions .....	228
<b>3.11.</b>	<b>CLA .....</b>	<b>274</b>
3.11.1.	Descriptions of Peripheral registers .....	275
3.11.2.	Descriptions of Peripheral functions .....	275
<b>3.12.</b>	<b>EXMC .....</b>	<b>288</b>
3.12.1.	Descriptions of Peripheral registers .....	288
3.12.2.	Descriptions of Peripheral functions .....	288
<b>3.13.</b>	<b>EXTI .....</b>	<b>296</b>
3.13.1.	Descriptions of Peripheral registers .....	296
3.13.2.	Descriptions of Peripheral functions .....	296
<b>3.14.</b>	<b>FAC .....</b>	<b>304</b>
3.14.1.	Descriptions of Peripheral registers .....	304
3.14.2.	Descriptions of Peripheral functions .....	305
<b>3.15.</b>	<b>FFT .....</b>	<b>323</b>
3.15.1.	Descriptions of Peripheral registers .....	323
3.15.2.	Descriptions of Peripheral functions .....	323
<b>3.16.</b>	<b>FMC .....</b>	<b>337</b>
3.16.1.	Descriptions of Peripheral registers .....	337
3.16.2.	Descriptions of Peripheral functions .....	338
<b>3.17.</b>	<b>FWDGT .....</b>	<b>374</b>
3.17.1.	Descriptions of Peripheral registers .....	374
3.17.2.	Descriptions of Peripheral functions .....	374
<b>3.18.</b>	<b>GPIO .....</b>	<b>379</b>
3.18.1.	Descriptions of Peripheral registers .....	379
3.18.2.	Descriptions of Peripheral functions .....	380
<b>3.19.</b>	<b>HPDF .....</b>	<b>391</b>
3.19.1.	Descriptions of Peripheral registers .....	392
3.19.2.	Descriptions of Peripheral functions .....	392
<b>3.20.</b>	<b>I2C .....</b>	<b>450</b>
3.20.1.	Descriptions of Peripheral registers .....	451

3.20.2.	Descriptions of Peripheral functions .....	451
<b>3.21.</b>	<b>LPTIMER .....</b>	<b>490</b>
3.21.1.	Descriptions of Peripheral registers .....	490
3.21.2.	Descriptions of Peripheral functions .....	491
<b>3.22.</b>	<b>MISC .....</b>	<b>511</b>
3.22.1.	Descriptions of Peripheral registers .....	511
3.22.2.	Descriptions of Peripheral functions .....	513
<b>3.23.</b>	<b>PMU .....</b>	<b>526</b>
3.23.1.	Descriptions of Peripheral registers .....	527
3.23.2.	Descriptions of Peripheral functions .....	527
<b>3.24.</b>	<b>QSPI .....</b>	<b>544</b>
3.24.1.	Descriptions of Peripheral registers .....	544
3.24.2.	Descriptions of Peripheral functions .....	544
<b>3.25.</b>	<b>RCU .....</b>	<b>564</b>
3.25.1.	Descriptions of Peripheral registers .....	564
3.25.2.	Descriptions of Peripheral functions .....	565
<b>3.26.</b>	<b>RTC .....</b>	<b>603</b>
3.26.1.	Descriptions of Peripheral registers .....	604
3.26.2.	Descriptions of Peripheral functions .....	605
<b>3.27.</b>	<b>HRTIMER .....</b>	<b>631</b>
3.27.1.	Descriptions of Peripheral registers .....	631
3.27.2.	Descriptions of Peripheral functions .....	633
<b>3.28.</b>	<b>SPI .....</b>	<b>714</b>
3.28.1.	Descriptions of Peripheral registers .....	714
3.28.2.	Descriptions of Peripheral functions .....	714
<b>3.29.</b>	<b>SYSCFG .....</b>	<b>740</b>
3.29.1.	Descriptions of Peripheral registers .....	740
3.29.2.	Descriptions of Peripheral functions .....	741
<b>3.30.</b>	<b>TIMER .....</b>	<b>766</b>
3.30.1.	Descriptions of Peripheral registers .....	766
3.30.2.	Descriptions of Peripheral functions .....	768
<b>3.31.</b>	<b>TMU .....</b>	<b>873</b>
3.31.1.	Descriptions of Peripheral registers .....	873
3.31.2.	Descriptions of Peripheral functions .....	873
<b>3.32.</b>	<b>TRIGSEL .....</b>	<b>887</b>
3.32.1.	Descriptions of Peripheral registers .....	888
3.32.2.	Descriptions of Peripheral functions .....	889
<b>3.33.</b>	<b>TRNG .....</b>	<b>898</b>
3.33.1.	Descriptions of Peripheral registers .....	899



3.33.2.	Descriptions of Peripheral functions .....	899
<b>3.34.</b>	<b>USART.....</b>	<b>921</b>
3.34.1.	Descriptions of Peripheral registers.....	921
3.34.2.	Descriptions of Peripheral functions .....	921
<b>3.35.</b>	<b>VREF .....</b>	<b>974</b>
3.35.1.	Descriptions of Peripheral registers.....	974
3.35.2.	Descriptions of Peripheral functions .....	974
<b>3.36.</b>	<b>WWDGT.....</b>	<b>979</b>
3.36.1.	Descriptions of Peripheral registers.....	979
3.36.2.	Descriptions of Peripheral functions .....	979
<b>4.</b>	<b>Revision history .....</b>	<b>984</b>

## List of Figures

Figure 2-1. File structure of firmware library of GD32G5x3 .....	42
Figure 2-2. Select peripheral example files .....	44
Figure 2-3. Copy the peripheral example files .....	44
Figure 2-4. Open the project file .....	44
Figure 2-5. Configure project files .....	45
Figure 2-6. Compile-debug-download .....	45

## List of Tables

Table 1-1. Peripherals .....	39
Table 2-1. Function descriptions of Firmware Library .....	46
Table 3-1. Peripheral function format of Firmware Library .....	47
Table 3-2. ADC Registers .....	47
Table 3-3. ADC firmware function.....	48
Table 3-4. Function adc_deinit.....	50
Table 3-5. Function adc_clock_config.....	50
Table 3-6. Function adc_special_function_config.....	52
Table 3-7. Function adc_data_alignment_config.....	53
Table 3-8. Function adc_enable .....	53
Table 3-9. Function adc_disable .....	54
Table 3-10. Function adc_calibration_mode_config .....	54
Table 3-11. Function adc_calibration_number .....	55
Table 3-12. Function adc_calibration_enable.....	56
Table 3-13. Function adc_resolution_config .....	56
Table 3-14. Function adc_internal_channel_config .....	57
Table 3-15. Function adc_gain_mode_enable .....	58
Table 3-16. Function adc_gain_mode_disable.....	58
Table 3-17. Function adc_gain_factor_set.....	59
Table 3-18. Function adc_dma_mode_enable.....	60
Table 3-19. Function adc_dma_mode_disable.....	60
Table 3-20. Function adc_dma_request_after_last_enable .....	61
Table 3-21. Function adc_dma_request_after_last_disable .....	61
Table 3-22. Function adc_hpdf_mode_enable .....	62
Table 3-23. Function adc_hpdf_mode_disable .....	62
Table 3-24. Function adc_discontinuous_mode_config .....	63
Table 3-25. Function adc_channel_length_config.....	63
Table 3-26. Function adc_routine_channel_config .....	64
Table 3-27. Function adc_inserted_channel_config .....	65
Table 3-28. Function adc_inserted_channel_offset_config.....	66
Table 3-29. Function adc_inserted_channel_mean_value_mode_enable.....	66
Table 3-30. Function adc_inserted_channel_mean_value_mode_disable.....	67
Table 3-31. Function adc_inserted_channel_mean_value_mode_config .....	67
Table 3-32. Function adc_channel_differential_mode_config .....	68
Table 3-33. Function adc_external_trigger_config.....	69
Table 3-34. Function adc_software_trigger_enable .....	70
Table 3-35. Function adc_end_of_conversion_config.....	70
Table 3-36. Function adc_routine_data_read .....	71
Table 3-37. Function adc_inserted_data_read .....	72
Table 3-38. Function adc_watchdog0_single_channel_enable .....	72

Table 3-39. Function <code>adc_watchdog0_sequence_channel_enable</code> .....	73
Table 3-40. Function <code>adc_watchdog0_disable</code> .....	74
Table 3-41. Function <code>adc_watchdog1_channel_config</code> .....	74
Table 3-42. Function <code>adc_watchdog2_channel_config</code> .....	75
Table 3-43. Function <code>adc_watchdog1_disable</code> .....	76
Table 3-44. Function <code>adc_watchdog2_disable</code> .....	76
Table 3-45. Function <code>adc_watchdog0_threshold_config</code> .....	77
Table 3-46. Function <code>adc_watchdog1_threshold_config</code> .....	77
Table 3-47. Function <code>adc_watchdog2_threshold_config</code> .....	78
Table 3-48. Function <code>adc_oversample_mode_config</code> .....	79
Table 3-49. Function <code>adc_oversample_mode_enable</code> .....	80
Table 3-50. Function <code>adc_oversample_mode_disable</code> .....	80
Table 3-51. Function <code>adc_flag_get</code> .....	81
Table 3-52. Function <code>adc_flag_clear</code> .....	82
Table 3-53. Function <code>adc_interrupt_enable</code> .....	83
Table 3-54. Function <code>adc_interrupt_disable</code> .....	83
Table 3-55. Function <code>adc_interrupt_flag_get</code> .....	84
Table 3-56. Function <code>adc_interrupt_flag_clear</code> .....	85
Table 3-57. Function <code>adc_sync_mode_config</code> .....	86
Table 3-58. Function <code>adc_interrupt_disable</code> .....	87
Table 3-59. Function <code>adc_sync_dma_config</code> .....	88
Table 3-60. Function <code>adc_sync_dma_request_after_last_enable</code> .....	88
Table 3-61. Function <code>adc_sync_dma_request_after_last_disable</code> .....	89
Table 3-62. Function <code>adc_sync_master_adc_routine_data_read</code> .....	89
Table 3-63. Function <code>adc_sync_slave_adc_routine_data_read</code> .....	90
Table 3-64. CAN Registers .....	91
Table 3-65. CAN firmware function .....	92
Table 3-66. Structure <code>can_error_counter_struct</code> .....	93
Table 3-67. Structure <code>can_parameter_struct</code> .....	93
Table 3-68. Structure <code>can_mailbox_descriptor_struct</code> .....	94
Table 3-69. Structure <code>can_rx_fifo_struct</code> .....	94
Table 3-70. Structure <code>can_fd_parameter_struct</code> .....	95
Table 3-71. Structure <code>can_rx_fifo_id_filter_struct</code> .....	95
Table 3-72. Structure <code>can_fifo_parameter_struct</code> .....	95
Table 3-73. Structure <code>can_pn_mode_filter_struct</code> .....	95
Table 3-74. Structure <code>can_pn_mode_config_struct</code> .....	96
Table 3-75. Structure <code>can_crc_struct</code> .....	96
Table 3-76. Enum <code>can_interrupt_enum</code> .....	96
Table 3-77. Enum <code>can_flag_enum</code> .....	98
Table 3-78. Enum <code>can_interrupt_flag_enum</code> .....	100
Table 3-79. Enum <code>can_operation_modes_enum</code> .....	102
Table 3-80. Enum <code>can_struct_type_enum</code> .....	103
Table 3-81. Enum <code>can_error_state_enum</code> .....	103
Table 3-82. Function <code>can_deinit</code> .....	103

Table 3-83. Function can_software_reset .....	104
Table 3-84. Function can_init .....	104
Table 3-85. Function can_struct_para_init .....	105
Table 3-86. Function can_private_filter_config .....	106
Table 3-87. Function can_operation_mode_enter .....	106
Table 3-88. Function can_operation_mode_get .....	107
Table 3-89. Function can_inactive_mode_exit .....	108
Table 3-90. Function can_pn_mode_exit .....	108
Table 3-91. Function can_fd_config .....	109
Table 3-92. Function can_bitrate_switch_enable .....	109
Table 3-93. Function can_bitrate_switch_disable .....	110
Table 3-94. Function can_tdc_get .....	110
Table 3-95. Function can_tdc_enable .....	111
Table 3-96. Function can_tdc_disable .....	111
Table 3-97. Function can_rx_fifo_config .....	112
Table 3-98. Function can_rx_fifo_filter_table_config .....	113
Table 3-99. Function can_rx_fifo_read .....	113
Table 3-100. Function can_rx_fifo_filter_matching_number_get .....	114
Table 3-101. Function can_rx_fifo_clear .....	114
Table 3-102. Function can_ram_address_get .....	115
Table 3-103. Function can_mailbox_config .....	115
Table 3-104. Function can_mailbox_transmit_abort .....	116
Table 3-105. Function can_mailbox_transmit_inactive .....	117
Table 3-106. Function can_mailbox_receive_data_read .....	117
Table 3-107. Function can_mailbox_receive_lock .....	118
Table 3-108. Function can_mailbox_receive_unlock .....	119
Table 3-109. Function can_mailbox_receive_inactive .....	119
Table 3-110. Function can_mailbox_code_get .....	120
Table 3-111. Function can_error_counter_config .....	120
Table 3-112. Function can_error_counter_get .....	121
Table 3-113. Function can_error_state_get .....	122
Table 3-114. Function can_crc_get .....	122
Table 3-115. Function can_pn_mode_config .....	123
Table 3-116. Function can_pn_mode_filter_config .....	123
Table 3-117. Function can_pn_mode_num_of_match_get .....	124
Table 3-118. Function can_pn_mode_data_read .....	125
Table 3-119. Function can_self_reception_enable .....	125
Table 3-120. Function can_self_reception_disable .....	126
Table 3-121. Function can_transmit_abort_enable .....	126
Table 3-122. Function can_transmit_abort_disable .....	127
Table 3-123. Function can_auto_busoff_recovery_enable .....	127
Table 3-124. Function can_auto_busoff_recovery_disable .....	128
Table 3-125. Function can_time_sync_enable .....	128
Table 3-126. Function can_time_sync_disable .....	129

Table 3-127. Function can_edge_filter_mode_enable .....	129
Table 3-128. Function can_edge_filter_mode_disable .....	130
Table 3-129. Function can_ped_mode_enable.....	130
Table 3-130. Function can_ped_mode_disable.....	131
Table 3-131. Function can_arbitration_delay_bits_config .....	131
Table 3-132. Function can_bsp_mode_config .....	132
Table 3-133. Function can_bsp_syn_config .....	132
Table 3-134. Function can_flag_get .....	133
Table 3-135. Function can_flag_clear .....	134
Table 3-136. Function can_interrupt_enable .....	134
Table 3-137. Function can_interrupt_disable .....	135
Table 3-138. Function can_interrupt_flag_get.....	135
Table 3-139. Function can_interrupt_flag_clear .....	136
Table 3-140. CAU Registers .....	137
Table 3-141. CAU firmware function .....	138
Table 3-142. Structure cau_key_parameter_struct.....	138
Table 3-143. Structure cau_iv_parameter_struct.....	139
Table 3-144. Structure cau_context_parameter_struct .....	139
Table 3-145. Structure cau_parameter_struct .....	139
Table 3-146. Function cau_deinit.....	140
Table 3-147. Function cau_struct_para_init .....	140
Table 3-148. Function cau_key_struct_para_init.....	141
Table 3-149. Function cau_iv_struct_para_init .....	141
Table 3-150. Function cau_context_struct_para_init.....	142
Table 3-151. Function cau_enable.....	142
Table 3-152. Function cau_disable.....	143
Table 3-153. Function cau_dma_enable.....	143
Table 3-154. Function cau_dma_disable .....	144
Table 3-155. Function cau_init .....	145
Table 3-156. Function cau_aes_keysize_config .....	146
Table 3-157. Function cau_key_init.....	146
Table 3-158. Function cau_iv_init.....	147
Table 3-159. Function cau_phase_config .....	148
Table 3-160. Function cau_fifo_flush.....	149
Table 3-161. Function cau_enable_state_get.....	149
Table 3-162. Function cau_data_write .....	150
Table 3-163. Function cau_data_read .....	150
Table 3-164. Function cau_context_save.....	151
Table 3-165. Function cau_context_restore .....	151
Table 3-166. Function cau_aes_ecb.....	152
Table 3-167. Function cau_aes_cbc.....	153
Table 3-168. Function cau_aes_ctr .....	154
Table 3-169. Function cau_aes_cfb.....	155
Table 3-170. Function cau_aes_ofb .....	156

Table 3-171. Function cau_aes_gcm.....	157
Table 3-172. Function cau_aes_ccm.....	158
Table 3-173. Function cau_tdes_ecb .....	159
Table 3-174. Function cau_tdes_cbc .....	160
Table 3-175. Function cau_des_ecb.....	161
Table 3-176. Function cau_des_cbc.....	161
Table 3-177. Function cau_flag_get .....	162
Table 3-178. Function cau_interrupt_enable .....	163
Table 3-179. Function cau_interrupt_disable .....	163
Table 3-180. Function cau_interrupt_flag_get.....	164
Table 3-181. CMP registers .....	165
Table 3-182. CMP firmware function .....	165
Table 3-183. Enum cmp_enum.....	166
Table 3-184. Function cmp_deinit .....	166
Table 3-185. Function cmp_mode_init.....	166
Table 3-186. Function cmp_noninverting_input_select .....	168
Table 3-187. Function cmp_output_init.....	168
Table 3-188. Function cmp_output_mux_config .....	169
Table 3-189. Function cmp_outputblank_init.....	170
Table 3-190. Function cmp_enable.....	171
Table 3-191. Function cmp_disable .....	171
Table 3-192. Function cmp_window_enable.....	172
Table 3-193. Function cmp_window_disable .....	172
Table 3-194. Function cmp_lock_enable.....	173
Table 3-195. Function cmp_voltage_scaler_enable .....	173
Table 3-196. Function cmp_voltage_scaler_disable .....	174
Table 3-197. Function cmp_scaler_bridge_enable.....	174
Table 3-198. Function cmp_scaler_bridge_disable.....	175
Table 3-199. Function cmp_output_level_get .....	175
Table 3-200. Function cmp_flag_get.....	176
Table 3-201. Function cmp_flag_clear .....	177
Table 3-202. Function cmp_interrupt_enable.....	177
Table 3-203. Function cmp_interrupt_disable.....	178
Table 3-204. Function cmp_interrupt_flag_get .....	178
Table 3-205. Function cmp_interrupt_flag_clear .....	179
Table 3-206. CPDM Registers .....	180
Table 3-207. CPDM firmware function .....	180
Table 3-208. Enum cpdm_output_phase_enum.....	180
Table 3-209. Function cpdm_enable .....	181
Table 3-210. Function cpdm_disable .....	182
Table 3-211. Function cpdm_delayline_sample_enable.....	182
Table 3-212. Function cpdm_delayline_sample_disable.....	183
Table 3-213. Function cpdm_output_clock_phase_select.....	183
Table 3-214. Function cpdm_delay_step_config .....	184

Table 3-215. Function cpdm_delayline_length_valid_flag_get .....	184
Table 3-216. Function cpdm_delayline_length_get.....	185
Table 3-217. Function cpdm_clock_output.....	185
Table 3-218. CRC Registers .....	186
Table 3-219. CRC firmware function .....	186
Table 3-220. Function crc_deinit .....	187
Table 3-221. Function crc_reverse_output_data_enable .....	187
Table 3-222. Function crc_reverse_output_data_disable .....	188
Table 3-223. Function crc_data_register_reset.....	188
Table 3-224. Function crc_data_register_read.....	189
Table 3-225. Function crc_free_data_register_read.....	189
Table 3-226. Function crc_free_data_register_write .....	190
Table 3-227. Function crc_init_data_register_write .....	190
Table 3-228. Function crc_input_data_reverse_config .....	191
Table 3-229. Function crc_polynomial_size_set.....	191
Table 3-230. Function crc_polynomial_set .....	192
Table 3-231. Function crc_single_data_calculate .....	192
Table 3-232. Function crc_block_data_calculate .....	193
Table 3-233. DAC Registers .....	194
Table 3-234. DAC firmware functions .....	195
Table 3-235. Function dac_deinit.....	196
Table 3-236. Function dac_enable .....	196
Table 3-237. Function dac_disable.....	197
Table 3-238. Function dac_dma_enable.....	198
Table 3-239. Function dac_dma_disable.....	198
Table 3-240. Function dac_mode_config .....	199
Table 3-241. Function dac_trimming_value_get.....	200
Table 3-242. Function dac_trimming_value_set .....	200
Table 3-243. Function dac_trimming_enable .....	201
Table 3-244. Function dac_trimming_disable .....	202
Table 3-245. Function dac_output_value_get .....	202
Table 3-246. Function dac_data_format_config.....	203
Table 3-247. Function dac_data_set .....	203
Table 3-248. Function dac_trigger_enable.....	204
Table 3-249. Function dac_trigger_disable.....	205
Table 3-250. Function dac_trigger_source_config.....	205
Table 3-251. Function dac_software_trigger_enable .....	206
Table 3-252. Function dac_wave_mode_config.....	207
Table 3-253. Function dac_lfsr_noise_config .....	208
Table 3-254. Function dac_triangle_noise_config.....	208
Table 3-255. Function dac_sawtooth_reset_trigger_source_config .....	209
Table 3-256. Function dac_sawtooth_step_trigger_source_config .....	210
Table 3-257. Function dac_sawtooth_step_direction_config.....	211
Table 3-258. Function dac_sawtooth_initial_value_set.....	211



Table 3-259. Function <code>dac_sawtooth_step_value_set</code> .....	212
Table 3-260. Function <code>dac_sawtooth_step_software_trigger_enable</code> .....	213
Table 3-261. Function <code>dac_concurrent_enable</code> .....	213
Table 3-262. Function <code>dac_concurrent_disable</code> .....	214
Table 3-263. Function <code>dac_concurrent_software_trigger_enable</code> .....	214
Table 3-264. Function <code>dac_concurrent_data_set</code> .....	215
Table 3-265. Function <code>dac_reset_persist_enable</code> .....	216
Table 3-266. Function <code>dac_reset_persist_disable</code> .....	216
Table 3-267. Function <code>dac_sample_keep_mode_config</code> .....	217
Table 3-268. Function <code>dac_flag_get</code> .....	218
Table 3-269. Function <code>dac_flag_clear</code> .....	218
Table 3-270. Function <code>dac_interrupt_enable</code> .....	219
Table 3-271. Function <code>dac_interrupt_disable</code> .....	219
Table 3-272. Function <code>dac_interrupt_flag_get</code> .....	220
Table 3-273. Function <code>dac_interrupt_flag_clear</code> .....	221
Table 3-274. DBG Registers .....	221
Table 3-275. DBG firmware function .....	222
Table 3-276. Enum <code>dbg_periph_enum</code> .....	222
Table 3-277. Function <code>dbg_deinit</code> .....	222
Table 3-278. Function <code>dbg_id_get</code> .....	223
Table 3-279. Function <code>dbg_low_power_enable</code> .....	223
Table 3-280. Function <code>dbg_low_power_disable</code> .....	224
Table 3-281. Function <code>dbg_trace_pin_enable</code> .....	225
Table 3-282. Function <code>dbg_trace_pin_disable</code> .....	225
Table 3-283. Function <code>dbg_periph_enable</code> .....	226
Table 3-284. Function <code>dbg_periph_disable</code> .....	226
Table 3-285. DMA Registers .....	227
Table 3-286. DMAMUX Registers .....	227
Table 3-287. DMA firmware function .....	228
Table 3-288. DMAMUX firmware function .....	228
Table 3-289. Structure <code>dma_parameter_struct</code> .....	229
Table 3-290. Structure <code>dmamux_sync_parameter_struct</code> .....	230
Table 3-291. Structure <code>dmamux_gen_parameter_struct</code> .....	230
Table 3-292. Enum <code>dma_channel_enum</code> .....	230
Table 3-293. Enum <code>dmamux_multiplexer_channel_enum</code> .....	230
Table 3-294. Enum <code>dmamux_generator_channel_enum</code> .....	231
Table 3-295. Enum <code>dmamux_interrupt_enum</code> .....	231
Table 3-296. Enum <code>dmamux_flag_enum</code> .....	232
Table 3-297. Enum <code>dmamux_interrupt_flag_enum</code> .....	233
Table 3-298. Function <code>dma_deinit</code> .....	234
Table 3-299. Function <code>dma_struct_para_init</code> .....	235
Table 3-300. Function <code>dma_init</code> .....	235
Table 3-301. Function <code>dma_circulation_enable</code> .....	236
Table 3-302. Function <code>dma_circulation_disable</code> .....	237

Table 3-303. Function dma_memory_to_memory_enable .....	237
Table 3-304. Function dma_memory_to_memory_disable .....	238
Table 3-305. Function dma_channel_enable .....	239
Table 3-306. Function dma_channel_disable .....	239
Table 3-307. Function dma_periph_address_config .....	240
Table 3-308. Function dma_memory_address_config .....	240
Table 3-309. Function dma_transfer_number_config .....	241
Table 3-310. Function dma_transfer_number_get .....	242
Table 3-311. Function dma_priority_config .....	242
Table 3-312. Function dma_memory_width_config .....	243
Table 3-313. Function dma_periph_width_config .....	244
Table 3-314. Function dma_memory_increase_enable .....	245
Table 3-315. Function dma_memory_increase_disable .....	245
Table 3-316. Function dma_periph_increase_enable .....	246
Table 3-317. Function dma_periph_increase_disable .....	246
Table 3-318. Function dma_transfer_direction_config .....	247
Table 3-319. Function dma_flag_get .....	248
Table 3-320. Function dma_flag_clear .....	248
Table 3-321. Function dma_interrupt_enable .....	249
Table 3-322. Function dma_interrupt_disable .....	250
Table 3-323. Function dma_interrupt_flag_get .....	250
Table 3-324. Function dma_interrupt_flag_clear .....	251
Table 3-325. Function dmamux_sync_struct_para_init .....	252
Table 3-326. Function dmamux_synchronization_init .....	253
Table 3-327. Function dmamux_synchronization_enable .....	253
Table 3-328. Function dmamux_synchronization_disable .....	254
Table 3-329. Function dmamux_event_generation_enable .....	254
Table 3-330. Function dmamux_event_generation_disable .....	255
Table 3-331. Function dmamux_gen_struct_para_init .....	256
Table 3-332. Function dmamux_request_generator_init .....	256
Table 3-333. Function dmamux_request_generator_channel_enable .....	257
Table 3-334. Function dmamux_request_generator_channel_disable .....	257
Table 3-335. Function dmamux_synchronization_polarity_config .....	258
Table 3-336. Function dmamux_request_forward_number_config .....	259
Table 3-337. Function dmamux_sync_id_config .....	259
Table 3-338. Function dmamux_request_id_config .....	261
Table 3-339. Function dmamux_trigger_polarity_config .....	268
Table 3-340. Function dmamux_request_generate_number_config .....	269
Table 3-341. Function dmamux_trigger_id_config .....	269
Table 3-342. Function dmamux_flag_get .....	271
Table 3-343. Function dmamux_flag_clear .....	272
Table 3-344. Function dmamux_interrupt_enable .....	272
Table 3-345. Function dmamux_interrupt_disable .....	273
Table 3-346. Function dmamux_interrupt_flag_get .....	273

Table 3-347. Function <code>dmamux_interrupt_flag_clear</code> .....	274
Table 3-348. CLA Registers .....	275
Table 3-349. CLA firmware function .....	275
Table 3-350. Function <code>cla_deinit</code> .....	276
Table 3-351. Function <code>cla_enable</code> .....	276
Table 3-352. Function <code>cla_disable</code> .....	277
Table 3-353. Function <code>cla_output_state_get</code> .....	277
Table 3-354. Function <code>cla_sigs_input_config</code> .....	278
Table 3-355. Function <code>cla_lcu_control_config</code> .....	279
Table 3-356. Function <code>cla_output_config</code> .....	279
Table 3-357. Function <code>cla_output_enable</code> .....	280
Table 3-358. Function <code>cla_output_disable</code> .....	280
Table 3-359. Function <code>cla_flip_flop_output_reset</code> .....	281
Table 3-360. Function <code>cla_flip_flop_clockpolarity_config</code> .....	281
Table 3-361. Function <code>cla_flip_flop_clocksource_config</code> .....	282
Table 3-362. Function <code>cla_flag_get</code> .....	283
Table 3-363. Function <code>cla_flag_clear</code> .....	283
Table 3-364. Function <code>cla_negedge_interrupt_enable</code> .....	284
Table 3-365. Function <code>cla_negedge_interrupt_disable</code> .....	284
Table 3-366. Function <code>cla_posedge_interrupt_enable</code> .....	285
Table 3-367. Function <code>cla_posedge_interrupt_disable</code> .....	286
Table 3-368. Function <code>cla_interrupt_flag_get</code> .....	286
Table 3-369. Function <code>cla_interrupt_flag_clear</code> .....	287
Table 3-370. EXMC Registers .....	288
Table 3-371. EXMC firmware function .....	288
Table 3-372. Structure <code>exmc_norsram_timing_parameter_struct</code> .....	288
Table 3-373. Structure <code>exmc_norsram_parameter_struct</code> .....	289
Table 3-374. Function <code>exmc_norsram_deinit</code> .....	289
Table 3-375. Function <code>exmc_norsram_struct_para_init</code> .....	290
Table 3-376. Function <code>exmc_norsram_init</code> .....	290
Table 3-377. Function <code>exmc_norsram_enable</code> .....	292
Table 3-378. Function <code>exmc_norsram_disable</code> .....	293
Table 3-379. Function <code>exmc_norsram_page_size_config</code> .....	293
Table 3-380. Function <code>exmc_norsram_consecutive_clock_config</code> .....	294
Table 3-381. Function <code>exmc_norsram_write_fifo_config</code> .....	294
Table 3-382. Function <code>exmc_norsram_status_get</code> .....	295
Table 3-383. EXTI Registers .....	296
Table 3-384. EXTI firmware function .....	296
Table 3-385. Enum <code>exti_line_enum</code> .....	297
Table 3-386. Enum <code>exti_mode_enum</code> .....	298
Table 3-387. Enum <code>exti_trig_type_enum</code> .....	298
Table 3-388. Function <code>exti_deinit</code> .....	298
Table 3-389. Function <code>exti_init</code> .....	299
Table 3-390. Function <code>exti_interrupt_enable</code> .....	299

Table 3-391. Function exti_interrupt_disable .....	300
Table 3-392. Function exti_event_enable .....	300
Table 3-393. Function exti_event_disable .....	301
Table 3-394. Function exti_software_interrupt_enable .....	301
Table 3-395. Function exti_software_interrupt_disable .....	302
Table 3-396. Function exti_flag_get .....	302
Table 3-397. Function exti_flag_clear .....	303
Table 3-398. Function exti_interrupt_flag_get .....	303
Table 3-399. Function exti_interrupt_flag_clear .....	304
Table 3-400. FAC Registers .....	304
Table 3-401. FAC firmware function .....	305
Table 3-402. Structure fac_parameter_struct .....	306
Table 3-403. Structure fac_fixed_data_preload_struct .....	306
Table 3-404. Structure fac_float_data_preload_struct .....	306
Table 3-405. Function fac_deinit .....	307
Table 3-406. Function fac_struct_para_init .....	307
Table 3-407. Function fac_fixed_data_preload_init .....	308
Table 3-408. Function fac_float_data_preload_init .....	308
Table 3-409. Function fac_init .....	309
Table 3-410. Function fac_preload .....	309
Table 3-411. Function fac_float_buffer_preload .....	310
Table 3-412. Function fac_float_preload .....	310
Table 3-413. Function fac_float_preload .....	311
Table 3-414. Function fac_init .....	312
Table 3-415. Function fac_clip_config .....	312
Table 3-416. Function fac_init .....	313
Table 3-417. Function fac_float_disable .....	313
Table 3-418. Function fac_dma_enable .....	314
Table 3-419. Function fac_dma_enable .....	314
Table 3-420. Function fac_dma_enable .....	315
Table 3-421. Function fac_dma_enable .....	315
Table 3-422. Function fac_dma_enable .....	316
Table 3-423. Function fac_dma_enable .....	316
Table 3-424. Function fac_start .....	317
Table 3-425. Function fac_start .....	317
Table 3-426. Function fac_start .....	318
Table 3-427. Function fac_fixed_data_write .....	318
Table 3-428. Function fac_fixed_data_write .....	319
Table 3-429. Function fac_fixed_data_write .....	319
Table 3-430. Function fac_fixed_data_write .....	320
Table 3-431. Function fac_interrupt_enable .....	320
Table 3-432. Function fac_interrupt_disable .....	321
Table 3-433. Function fac_interrupt_flag_get .....	322
Table 3-434. Function fac_flag_get .....	322

Table 3-435. FFT Registers .....	323
Table 3-436. FFT firmware function.....	324
Table 3-437. Structure fft_parameter_struct.....	324
Table 3-438. Function fft_deinit .....	325
Table 3-439. Function fft_struct_para_init .....	325
Table 3-440. Function fft_init.....	326
Table 3-441. Function fft_calculation_start .....	327
Table 3-442. Function fft_calculation_stop.....	327
Table 3-443. Function fft_point_number_config.....	328
Table 3-444. Function fft_mode_config .....	328
Table 3-445. Function fft_window_enable .....	329
Table 3-446. Function fft_window_disable .....	329
Table 3-447. Function fft_downsample_config .....	330
Table 3-448. Function fft_image_source_config.....	330
Table 3-449. Function fft_real_addr_config .....	331
Table 3-450. Function fft_image_addr_config.....	331
Table 3-451. Function fft_window_addr_config.....	332
Table 3-452. Function fft_output_addr_config .....	332
Table 3-453. Function fft_loop_buffer_length_config.....	333
Table 3-454. Function fft_loop_buffer_index_config .....	333
Table 3-455. Function fft_flag_get .....	334
Table 3-456. Function fft_flag_clear.....	335
Table 3-457. Function fft_interrupt_enable.....	335
Table 3-458. Function fft_interrupt_disable .....	336
Table 3-459. Function fft_interrupt_flag_get .....	336
Table 3-460. Function fft_interrupt_flag_clear .....	337
Table 3-461. FMC Registers .....	337
Table 3-462. FMC firmware function .....	338
Table 3-463. Enum fmc_state_enum .....	340
Table 3-464. Function fmc_unlock .....	340
Table 3-465. Function fmc_lock .....	341
Table 3-466. Function fmc_wsnt_set .....	341
Table 3-467. Function fmc_prefetch_enable .....	342
Table 3-468. Function fmc_prefetch_disable .....	342
Table 3-469. Function fmc_icache_enable.....	343
Table 3-470. Function fmc_icache_disable.....	343
Table 3-471. Function fmc_icache_reset_enable .....	344
Table 3-472. Function fmc_dcache_enable .....	344
Table 3-473. Function fmc_dcache_disable .....	345
Table 3-474. Function fmc_dcache_reset_enable .....	345
Table 3-475. Function fmc_page_erase.....	346
Table 3-476. Function fmc_bank0_erase.....	346
Table 3-477. Function fmc_bank1_erase.....	347
Table 3-478. Function fmc_mass_erase.....	347

Table 3-479. Function <code>fmc_doubleword_program</code> .....	348
Table 3-480. Function <code>fmc_bank0_base_address_get</code> .....	348
Table 3-481. Function <code>fmc_bank1_base_address_get</code> .....	349
Table 3-482. Function <code>fmc_page_size_get</code> .....	349
Table 3-483. Function <code>fmc_debugger_enable</code> .....	350
Table 3-484. Function <code>fmc_debugger_disable</code> .....	350
Table 3-485. Function <code>fmc_slp_pd_mode_enable</code> .....	351
Table 3-486. Function <code>fmc_slp_pd_mode_disable</code> .....	351
Table 3-487. Function <code>fmc_scr_area_enable</code> .....	352
Table 3-488. Function <code>ob_unlock</code> .....	352
Table 3-489. Function <code>ob_lock</code> .....	353
Table 3-490. Function <code>ob_reload</code> .....	354
Table 3-491. Function <code>ob_user_write</code> .....	354
Table 3-492. Function <code>ob_security_protection_config</code> .....	357
Table 3-493. Function <code>ob_dcrp_config</code> .....	357
Table 3-494. Function <code>ob_write_protection_config</code> .....	358
Table 3-495. Function <code>ob_scr_area_config</code> .....	359
Table 3-496. Function <code>ob_boot_lock_config</code> .....	360
Table 3-497. Function <code>ob_bank_memory_swap_config</code> .....	360
Table 3-498. Function <code>ob_user_get</code> .....	361
Table 3-499. Function <code>ob_security_protection_level_get</code> .....	361
Table 3-500. Function <code>ob_dcrp_area_get</code> .....	362
Table 3-501. Function <code>ob_write_protection_get</code> .....	363
Table 3-502. Function <code>ob_scr_area_size_get</code> .....	364
Table 3-503. Function <code>ob_boot_config_get</code> .....	364
Table 3-504. Function <code>fmc_flag_get</code> .....	365
Table 3-505. Function <code>fmc_flag_clear</code> .....	366
Table 3-506. Function <code>fmc_interrupt_enable</code> .....	366
Table 3-507. Function <code>fmc_interrupt_disable</code> .....	367
Table 3-508. Function <code>fmc_ecc_flag_get</code> .....	367
Table 3-509. Function <code>fmc_ecc_flag_clear</code> .....	368
Table 3-510. Function <code>fmc_eccor_interrupt_enable</code> .....	369
Table 3-511. Function <code>fmc_eccor_interrupt_disable</code> .....	369
Table 3-512. Function <code>fmc_eccor_interrupt_flag_get</code> .....	370
Table 3-513. Function <code>fmc_eccor_interrupt_flag_clear</code> .....	370
Table 3-514. Function <code>fmc_interrupt_flag_get</code> .....	371
Table 3-515. Function <code>fmc_interrupt_flag_clear</code> .....	371
Table 3-516. Function <code>fmc_pd_mode_enter</code> .....	372
Table 3-517. Function <code>fmc_pd_mode_exit</code> .....	373
Table 3-518. Function <code>ob_bank_mode_config</code> .....	373
Table 3-519. FWDGT Registers .....	374
Table 3-520. FWDGT firmware function.....	374
Table 3-521. Function <code>fwdgt_write_enable</code> .....	375
Table 3-522. Function <code>fwdgt_write_disable</code> .....	375

Table 3-523. Function fwdgt_enable .....	376
Table 3-524. Function fwdgt_prescaler_value_config .....	376
Table 3-525. Function fwdgt_reload_value_config.....	377
Table 3-526. Function fwdgt_window_value_config .....	377
Table 3-527. Function fwdgt_counter_reload.....	378
Table 3-528. Function fwdgt_config.....	378
Table 3-529. Function fwdgt_flag_get.....	379
Table 3-530. GPIO Registers.....	380
Table 3-531. GPIO firmware function .....	380
Table 3-532. Function gpio_deinit .....	381
Table 3-533. Function gpio_mode_set.....	381
Table 3-534. Function gpio_output_options_set .....	382
Table 3-535. Function gpio_bit_set .....	383
Table 3-536. Function gpio_bit_reset .....	384
Table 3-537. Function gpio_bit_write.....	384
Table 3-538. Function gpio_port_write .....	385
Table 3-539. Function gpio_input_filter_set .....	385
Table 3-540. Function gpio_input_bit_get.....	386
Table 3-541. Function gpio_input_port_get.....	387
Table 3-542. Function gpio_output_bit_get .....	387
Table 3-543. Function gpio_output_port_get .....	388
Table 3-544. Function gpio_af_set .....	389
Table 3-545. Function gpio_pin_lock .....	390
Table 3-546. Function gpio_bit_toggle .....	390
Table 3-547. Function gpio_port_toggle .....	391
Table 3-548. HPDF Registers.....	392
Table 3-549. HPDF firmware function .....	392
Table 3-550. Structure hpdf_channel_parameter_struct.....	395
Table 3-551. Structure hpdf_filter_parameter_struct.....	395
Table 3-552. Structure hpdf_rc_parameter_struct.....	396
Table 3-553. Structure hpdf_ic_parameter_struct .....	396
Table 3-554. Enum hpdf_channel_enum .....	396
Table 3-555. Enum hpdf_filter_enum .....	396
Table 3-556. Enum hpdf_flag_enum.....	397
Table 3-557. Enum hpdf_interrput_flag_enum.....	399
Table 3-558. Enum hpdf_interrput_enum.....	400
Table 3-559. Function hpdf_deinit .....	400
Table 3-560. Function hpdf_channel_struct_para_init .....	401
Table 3-561. Function hpdf_filter_struct_para_init .....	401
Table 3-562. Function hpdf_rc_struct_para_init .....	402
Table 3-563. Function hpdf_ic_struct_para_init .....	402
Table 3-564. Function hpdf_enable .....	403
Table 3-565. Function hpdf_disable .....	403
Table 3-566. Function hpdf_channel_init .....	404



Table 3-567. Function <code>hpdf_filter_init</code> .....	405
Table 3-568. Function <code>hpdf_rc_init</code> .....	406
Table 3-569. Function <code>hpdf_ic_init</code> .....	406
Table 3-570. Function <code>hpdf_clock_output_config</code> .....	407
Table 3-571. Function <code>hpdf_clock_output_source_config</code> .....	408
Table 3-572. Function <code>hpdf_clock_output_duty_mode_disable</code> .....	408
Table 3-573. Function <code>hpdf_clock_output_duty_mode_enable</code> .....	409
Table 3-574. Function <code>hpdf_clock_output_divider_config</code> .....	409
Table 3-575. Function <code>hpdf_channel_enable</code> .....	410
Table 3-576. Function <code>hpdf_channel_disable</code> .....	410
Table 3-577. Function <code>hpdf_spi_clock_source_config</code> .....	411
Table 3-578. Function <code>hpdf_serial_interface_type_config</code> .....	412
Table 3-579. Function <code>hpdf_malfunction_monitor_disable</code> .....	412
Table 3-580. Function <code>hpdf_malfunction_monitor_enable</code> .....	413
Table 3-581. Function <code>hpdf_clock_loss_disable</code> .....	413
Table 3-582. Function <code>hpdf_clock_loss_enable</code> .....	414
Table 3-583. Function <code>hpdf_channel_pin_redirection_disable</code> .....	414
Table 3-584. Function <code>hpdf_channel_pin_redirection_enable</code> .....	415
Table 3-585. Function <code>hpdf_channel_multiplexer_config</code> .....	415
Table 3-586. Function <code>hpdf_data_pack_mode_config</code> .....	416
Table 3-587. Function <code>hpdf_data_right_bit_shift_config</code> .....	417
Table 3-588. Function <code>hpdf_calibration_offset_config</code> .....	417
Table 3-589. Function <code>hpdf_malfunction_break_signal_config</code> .....	418
Table 3-590. Function <code>hpdf_malfunction_counter_config</code> .....	419
Table 3-591. Function <code>hpdf_write_parallel_data_standard_mode</code> .....	419
Table 3-592. Function <code>hpdf_write_parallel_data_interleaved_mode</code> .....	420
Table 3-593. Function <code>hpdf_write_parallel_data_dual_mode</code> .....	420
Table 3-594. Function <code>hpdf_pulse_skip_update</code> .....	421
Table 3-595. Function <code>hpdf_pulse_skip_read</code> .....	421
Table 3-596. Function <code>hpdf_filter_enable</code> .....	422
Table 3-597. Function <code>hpdf_filter_disable</code> .....	422
Table 3-598. Function <code>hpdf_filter_config</code> .....	423
Table 3-599. Function <code>hpdf_integrator_oversample</code> .....	424
Table 3-600. Function <code>hpdf_threshold_monitor_filter_config</code> .....	424
Table 3-601. Function <code>hpdf_threshold_monitor_filter_read_data</code> .....	425
Table 3-602. Function <code>hpdf_threshold_monitor_fast_mode_disable</code> .....	426
Table 3-603. Function <code>hpdf_threshold_monitor_fast_mode_enable</code> .....	426
Table 3-604. Function <code>hpdf_threshold_monitor_channel</code> .....	427
Table 3-605. Function <code>hpdf_threshold_monitor_high_threshold</code> .....	427
Table 3-606. Function <code>hpdf_threshold_monitor_low_threshold</code> .....	428
Table 3-607. Function <code>hpdf_high_threshold_break_signal</code> .....	428
Table 3-608. Function <code>hpdf_low_threshold_break_signal</code> .....	429
Table 3-609. Function <code>hpdf_extremes_monitor_channel</code> .....	430
Table 3-610. Function <code>hpdf_extremes_monitor_maximum_get</code> .....	430



Table 3-611. Function <code>hpdf_extremes_monitor_minimum_get</code> .....	431
Table 3-612. Function <code>hpdf_conversion_time_get</code> .....	431
Table 3-613. Function <code>hpdf_rc_continuous_disable</code> .....	432
Table 3-614. Function <code>hpdf_rc_continuous_enable</code> .....	432
Table 3-615. Function <code>hpdf_rc_start_by_software</code> .....	433
Table 3-616. Function <code>hpdf_rc_syn_disable</code> .....	433
Table 3-617. Function <code>hpdf_rc_syn_enable</code> .....	434
Table 3-618. Function <code>hpdf_rc_dma_disable</code> .....	434
Table 3-619. Function <code>hpdf_rc_dma_enable</code> .....	435
Table 3-620. Function <code>hpdf_rc_channel_config</code> .....	435
Table 3-621. Function <code>hpdf_rc_fast_mode_disable</code> .....	436
Table 3-622. Function <code>hpdf_rc_fast_mode_enable</code> .....	437
Table 3-623. Function <code>hpdf_rc_data_get</code> .....	437
Table 3-624. Function <code>hpdf_rc_channel_get</code> .....	438
Table 3-625. Function <code>hpdf_ic_start_by_software</code> .....	438
Table 3-626. Function <code>hpdf_ic_syn_disable</code> .....	439
Table 3-627. Function <code>hpdf_ic_syn_enable</code> .....	439
Table 3-628. Function <code>hpdf_ic_dma_disable</code> .....	440
Table 3-629. Function <code>hpdf_ic_dma_enable</code> .....	440
Table 3-630. Function <code>hpdf_ic_scan_mode_disable</code> .....	441
Table 3-631. Function <code>hpdf_ic_scan_mode_enable</code> .....	441
Table 3-632. Function <code>hpdf_ic_trigger_signal_disable</code> .....	442
Table 3-633. Function <code>hpdf_ic_trigger_signal_config</code> .....	442
Table 3-634. Function <code>hpdf_ic_channel_config</code> .....	443
Table 3-635. Function <code>hpdf_ic_data_get</code> .....	444
Table 3-636. Function <code>hpdf_ic_channel_get</code> .....	444
Table 3-637. Function <code>hpdf_flag_get</code> .....	445
Table 3-638. Function <code>hpdf_flag_clear</code> .....	446
Table 3-639. Function <code>hpdf_interrupt_enable</code> .....	447
Table 3-640. Function <code>hpdf_interrupt_disable</code> .....	448
Table 3-641. Function <code>hpdf_interrupt_flag_get</code> .....	449
Table 3-642. Function <code>hpdf_interrupt_flag_clear</code> .....	450
Table 3-643. I2C Registers .....	451
Table 3-644. I2C firmware function.....	451
Table 3-645. Enum <code>i2c_interrupt_flag_enum</code> .....	453
Table 3-646. Function <code>i2c_deinit</code> .....	453
Table 3-647. Function <code>i2c_timing_config</code> .....	454
Table 3-648. Function <code>i2c_digital_noise_filter_config</code> .....	455
Table 3-649. Function <code>i2c_analog_noise_filter_enable</code> .....	456
Table 3-650. Function <code>i2c_analog_noise_filter_disable</code> .....	456
Table 3-651. Function <code>i2c_master_clock_config</code> .....	457
Table 3-652. Function <code>i2c_master_addressing</code> .....	457
Table 3-653. Function <code>i2c_address10_header_enable</code> .....	458
Table 3-654. Function <code>i2c_address10_header_disable</code> .....	458

Table 3-655. Function i2c_address10_enable .....	459
Table 3-656. Function i2c_address10_disable .....	459
Table 3-657. Function i2c_automatic_end_enable .....	460
Table 3-658. Function i2c_automatic_end_disable .....	460
Table 3-659. Function i2c_slave_response_to_gcall_enable .....	461
Table 3-660. Function i2c_slave_response_to_gcall_disable .....	462
Table 3-661. Function i2c_stretch_scl_low_enable .....	462
Table 3-662. Function i2c_stretch_scl_low_disable .....	463
Table 3-663. Function i2c_address_config .....	463
Table 3-664. Function i2c_address_bit_compare_config .....	464
Table 3-665. Function i2c_address_disable .....	465
Table 3-666. Function i2c_second_address_config.....	465
Table 3-667. Function i2c_second_address_disable .....	466
Table 3-668. Function i2c_receved_address_get .....	467
Table 3-669. Function i2c_slave_byte_control_enable.....	467
Table 3-670. Function i2c_slave_byte_control_disable.....	468
Table 3-671. Function i2c_nack_enable .....	468
Table 3-672. Function i2c_nack_disable .....	469
Table 3-673. Function i2c_wakeup_from_deepsleep_enable.....	469
Table 3-674. Function i2c_wakeup_from_deepsleep_disable.....	470
Table 3-675. Function i2c_enable .....	470
Table 3-676. Function i2c_disable .....	471
Table 3-677. Function i2c_start_on_bus .....	471
Table 3-678. Function i2c_stop_on_bus .....	472
Table 3-679. Function i2c_data_transmit .....	472
Table 3-680. Function i2c_data_receive .....	473
Table 3-681. Function i2c_reload_enable.....	473
Table 3-682. Function i2c_reload_disable .....	474
Table 3-683. Function i2c_transfer_byte_number_config.....	474
Table 3-684. Function i2c_dma_enable .....	475
Table 3-685. Function i2c_dma_disable .....	476
Table 3-686. Function i2c_pec_transfer .....	476
Table 3-687. Function i2c_pec_enable .....	477
Table 3-688. Function i2c_pec_disable .....	477
Table 3-689. Function i2c_pec_value_get.....	478
Table 3-690. Function i2c_smbus_alert_enable.....	478
Table 3-691. Function i2c_smbus_alert_disable.....	479
Table 3-692. Function i2c_smbus_default_addr_enable.....	479
Table 3-693. Function i2c_smbus_default_addr_disable.....	480
Table 3-694. Function i2c_smbus_host_addr_enable .....	480
Table 3-695. Function i2c_smbus_host_addr_disable .....	481
Table 3-696. Function i2c_extented_clock_timeout_enable.....	481
Table 3-697. Function i2c_extented_clock_timeout_disable.....	482
Table 3-698. Function i2c_clock_timeout_enable .....	482

Table 3-699. Function i2c_clock_timeout_disable .....	483
Table 3-700. Function i2c_bus_timeout_b_config.....	483
Table 3-701. Function i2c_bus_timeout_a_config .....	484
Table 3-702. Function i2c_idle_clock_timeout_config .....	484
Table 3-703. Function i2c_flag_get .....	485
Table 3-704. Function i2c_flag_clear .....	486
Table 3-705. Function i2c_interrupt_enable .....	487
Table 3-706. Function i2c_interrupt_disable .....	487
Table 3-707. Function i2c_interrupt_flag_get.....	488
Table 3-708. Function i2c_interrupt_flag_clear.....	489
Table 3-709. LPTIMER Registers .....	490
Table 3-710. LPTIMER firmware function .....	491
Table 3-711. Structure lptimer_parameter_struct .....	492
Table 3-712. Function lptimer_deinit.....	492
Table 3-713. Function lptimer_struct_para_init .....	493
Table 3-714. Function lptimer_init .....	493
Table 3-715. Function lptimer_inputremap .....	494
Table 3-716. Function lptimer_register_shadow_enable .....	495
Table 3-717. Function lptimer_register_shadow_disable .....	496
Table 3-718. Function lptimer_timeout_enable.....	496
Table 3-719. Function lptimer_timeout_disable.....	497
Table 3-720. Function lptimer_timeout_enable.....	497
Table 3-721. Function lptimer_timeout_enable.....	498
Table 3-722. Function lptimer_timeout_enable.....	498
Table 3-723. Function lptimer_continue_start .....	499
Table 3-724. Function lptimer_single_start .....	499
Table 3-725. Function lptimer_stop.....	500
Table 3-726. Function lptimer_counter_read.....	500
Table 3-727. Function lptimer_autoreload_read .....	501
Table 3-728. Function lptimer_compare_read.....	501
Table 3-729. Function lptimer_autoreload_value_config .....	502
Table 3-730. Function lptimer_compare_value_config.....	502
Table 3-731. Function lptimer_decodemode0_enable .....	503
Table 3-732. Function lptimer_decodemode1_enable .....	503
Table 3-733. Function lptimer_decodemode_disable .....	504
Table 3-734. Function lptimer_highlevelcounter_enable .....	504
Table 3-735. Function lptimer_highlevelcounter_disable .....	505
Table 3-736. Function lptimer_flag_get .....	505
Table 3-737. Function lptimer_flag_clear .....	506
Table 3-738. Function lptimer_interrupt_enable .....	507
Table 3-739. Function lptimer_interrupt_disable .....	508
Table 3-740. Function lptimer_interrupt_flag_get .....	509
Table 3-741. Function lptimer_interrupt_flag_clear .....	510
Table 3-742. NVIC Registers .....	511

Table 3-743. SysTick Registers .....	513
Table 3-744. MISC firmware function .....	513
Table 3-745. Structure mpu_region_init_struct .....	514
Table 3-746. Structure mpu_attribute_init_struct.....	514
Table 3-747. Enum IRQn_Type .....	514
Table 3-748. Function nvic_priority_group_set .....	518
Table 3-749. Function nvic_irq_enable.....	518
Table 3-750. Function nvic_irq_disable.....	519
Table 3-751. Function nvic_system_reset .....	519
Table 3-752. Function nvic_vector_table_set.....	520
Table 3-753. Function system_lowpower_set .....	520
Table 3-754. Function system_lowpower_reset.....	521
Table 3-755. Function systick_clksource_set .....	522
Table 3-756. Function mpu_enable .....	522
Table 3-757. Function mpu_disable .....	523
Table 3-758. Function mpu_region_struct_para_init .....	523
Table 3-759. Function mpu_attribute_struct_para_init.....	524
Table 3-760. Function mpu_region_config .....	524
Table 3-761. Function mpu_attribute_config.....	525
Table 3-762. Function mpu_region_enable.....	526
Table 3-763. PMU Registers.....	527
Table 3-764. PMU firmware function .....	527
Table 3-765. Function pmu_deinit .....	528
Table 3-766. Function pmu_lvd_select .....	528
Table 3-767. Function pmu_lvd_enable.....	529
Table 3-768. Function pmu_lvd_disable.....	529
Table 3-769. Function pmu_avd_select .....	530
Table 3-770. Function pmu_avd_enable.....	531
Table 3-771. Function pmu_avd_disable.....	531
Table 3-772. Function pmu_ovd_enable .....	532
Table 3-773. Function pmu_ovd_enable .....	532
Table 3-774. Function pmu_ovd_disable .....	533
Table 3-775. Function pmu_uvd_select.....	533
Table 3-776. Function pmu_uvd_enable .....	534
Table 3-777. Function pmu_uvd_disable .....	534
Table 3-778. Function pmu_ovd_filter .....	535
Table 3-779. Function pmu_uvd_filter .....	535
Table 3-780. Function pmu_deepsleep_voltage .....	536
Table 3-781. Function pmu_vbat_charging_select.....	536
Table 3-782. Function pmu_vbat_charging_enable .....	537
Table 3-783. Function pmu_vbat_charging_disable .....	537
Table 3-784. Function pmu_vbat_temp_monitor_enable .....	538
Table 3-785. Function pmu_vbat_temp_monitor_disable .....	538
Table 3-786. Function pmu_to_sleepmode.....	539

Table 3-787. Function pmu_to_deepsleepmode .....	539
Table 3-788. Function pmu_to_standbymode .....	540
Table 3-789. Function pmu_wakeup_pin_enable .....	540
Table 3-790. Function pmu_wakeup_pin_disable .....	541
Table 3-791. Function pmu_backup_write_enable .....	542
Table 3-792. Function pmu_backup_write_disable .....	542
Table 3-793. Function pmu_flag_get .....	543
Table 3-794. Function pmu_flag_clear .....	543
Table 3-795. QSPI registers .....	544
Table 3-796. QSPI firmware function .....	545
Table 3-797. Structure qspi_init_struct .....	545
Table 3-798. Structure qspi_command_struct .....	546
Table 3-799. Structure qspi_polling_struct .....	546
Table 3-800. Function qspi_deinit .....	546
Table 3-801. Function qspi_struct_para_init .....	547
Table 3-802. Function qspi_cmd_struct_para_init .....	547
Table 3-803. Function qspi_polling_struct_para_init .....	548
Table 3-804. Function qspi_init .....	549
Table 3-805. Function qspi_enable .....	549
Table 3-806. Function qspi_disable .....	550
Table 3-807. Function qspi_dma_enable .....	550
Table 3-808. Function qspi_dma_disable .....	551
Table 3-809. Function qspi_command_config .....	551
Table 3-810. Function qspi_polling_config .....	552
Table 3-811. Function qspi_memorymapped_config .....	553
Table 3-812. Function qspi_data_transmit .....	554
Table 3-813. Function qspi_data_receive .....	555
Table 3-814. Function qspi_transmission_abort .....	555
Table 3-815. Function qspi_output_clock_delay_enable .....	556
Table 3-816. Function qspi_output_clock_delay_disable .....	556
Table 3-817. Function qspi_output_clock_delay_config .....	557
Table 3-818. Function qspi_sample_shift_config .....	557
Table 3-819. Function qspi_receive_clock_sel .....	558
Table 3-820. Function qspi_delay_scan_enable .....	559
Table 3-821. Function qspi_delay_scan_disable .....	559
Table 3-822. Function qspi_csn_edge_cycle .....	560
Table 3-823. Function qspi_flag_get .....	560
Table 3-824. Function qspi_flag_clear .....	561
Table 3-825. Function qspi_interrupt_enable .....	562
Table 3-826. Function qspi_interrupt_disable .....	562
Table 3-827. Function qspi_interrupt_flag_get .....	563
Table 3-828. Function qspi_interrupt_flag_clear .....	564
Table 3-829. RCU Registers .....	564
Table 3-830. RCU firmware function .....	565

Table 3-831. Enum rcu_periph_enum .....	567
Table 3-832. Enum rcu_periph_sleep_enum .....	568
Table 3-833. Enum rcu_periph_reset_enum .....	570
Table 3-834. Enum rcu_flag_enum .....	572
Table 3-835. Enum rcu_int_flag_enum .....	572
Table 3-836. Enum rcu_int_flag_clear_enum .....	573
Table 3-837. Enum rcu_int_enum .....	573
Table 3-838. Enum rcu_osci_type_enum .....	574
Table 3-839. Enum rcu_clock_freq_enum .....	574
Table 3-840. Enum usart_idx_enum .....	574
Table 3-841. Enum i2c_idx_enum .....	574
Table 3-842. Enum can_idx_enum .....	575
Table 3-843. Enum adc_idx_enum .....	575
Table 3-844. Function rcu_deinit .....	575
Table 3-845. Function rcu_periph_clock_enable .....	575
Table 3-846. Function rcu_periph_clock_disable .....	576
Table 3-847. Function rcu_periph_clock_sleep_enable .....	576
Table 3-848. Function rcu_periph_clock_sleep_disable .....	577
Table 3-849. Function rcu_periph_reset_enable .....	577
Table 3-850. Function rcu_periph_reset_disable .....	578
Table 3-851. Function rcu_bkp_reset_enable .....	578
Table 3-852. Function rcu_bkp_reset_disable .....	579
Table 3-853. Function rcu_system_clock_source_config .....	579
Table 3-854. Function rcu_system_clock_source_get .....	580
Table 3-855. Function rcu_ahb_clock_config .....	581
Table 3-856. Function rcu_apb1_clock_config .....	581
Table 3-857. Function rcu_apb2_clock_config .....	582
Table 3-858. Function rcu_apb3_clock_config .....	582
Table 3-859. Function rcu_ckout_config .....	583
Table 3-860. Function rcu_lsckout_enable .....	583
Table 3-861. Function rcu_lsckout_disable .....	584
Table 3-862. Function rcu_lsckout_config .....	584
Table 3-863. Function rcu_pll_source_config .....	585
Table 3-864. Function rcu_pll_config .....	586
Table 3-865. Function rcu_pll_clock_output_enable .....	586
Table 3-866. Function rcu_pll_clock_output_disable .....	587
Table 3-867. Function rcu_rtc_clock_config .....	588
Table 3-868. Function rcu_usart_clock_config .....	588
Table 3-869. Function rcu_i2c_clock_config .....	589
Table 3-870. Function rcu_can_clock_config .....	589
Table 3-871. Function rcu_adc_clock_config .....	590
Table 3-872. Function rcu_hpdf_clock_config .....	591
Table 3-873. Function rcu_hpdfaudio_clock_config .....	591
Table 3-874. Function rcu_trng_clock_config .....	592

Table 3-875. Function rcu_lptimer_clock_config .....	592
Table 3-876. Function rcu_qspi_clock_config .....	593
Table 3-877. Function rcu_hrtimer_clock_config .....	594
Table 3-878. Function rcu_lxtal_drive_capability_config.....	594
Table 3-879. Function rcu_osci_stab_wait .....	595
Table 3-880. Function rcu_osci_on .....	595
Table 3-881. Function rcu_osci_off.....	596
Table 3-882. Function rcu_osci_bypass_mode_enable .....	596
Table 3-883. Function rcu_osci_bypass_mode_disable .....	597
Table 3-884. Function rcu_irc8m_adjust_value_set.....	597
Table 3-885. Function rcu_hxtal_clock_monitor_enable .....	598
Table 3-886. Function rcu_hxtal_clock_monitor_disable .....	598
Table 3-887. Function rcu_lxtal_clock_monitor_enable.....	599
Table 3-888. Function rcu_lxtal_clock_monitor_disable.....	599
Table 3-889. Function rcu_clock_freq_get.....	600
Table 3-890. Function rcu_flag_get.....	600
Table 3-891. Function rcu_all_reset_flag_clear .....	601
Table 3-892. Function rcu_interrupt_enable.....	601
Table 3-893. Function rcu_interrupt_disable.....	602
Table 3-894. Function rcu_interrupt_flag_get .....	602
Table 3-895. Function rcu_interrupt_flag_clear .....	603
Table 3-896. RTC Registers .....	604
Table 3-897. RTC firmware function.....	605
Table 3-898. Structure rtc_parameter_struct.....	606
Table 3-899. Structure rtc_alarm_struct.....	606
Table 3-900. Structure rtc_timestamp_struct.....	607
Table 3-901. Structure rtc_tamper_struct .....	607
Table 3-902. Function rtc_deinit .....	607
Table 3-903. Function rtc_init.....	608
Table 3-904. Function rtc_init_mode_enter .....	609
Table 3-905. Function rtc_init_mode_exit.....	609
Table 3-906. Function rtc_register_sync_wait .....	610
Table 3-907. Function rtc_current_time_get.....	610
Table 3-908. Function rtc_subsecond_get.....	611
Table 3-909. Function rtc_alarm_config.....	611
Table 3-910. Function rtc_alarm_subsecond_config.....	612
Table 3-911. Function rtc_alarm_enable .....	613
Table 3-912. Function rtc_alarm_disable .....	614
Table 3-913. Function rtc_alarm_get.....	614
Table 3-914. Function rtc_alarm_subsecond_get .....	615
Table 3-915. Function rtc_timestamp_enable .....	615
Table 3-916. Function rtc_timestamp_disable .....	616
Table 3-917. Function rtc_timestamp_internalevent_config .....	616
Table 3-918. Function rtc_timestamp_get.....	617



Table 3-919. Function rtc_timestamp_subsecond_get.....	617
Table 3-920. Function rtc_tamper_enable.....	618
Table 3-921. Function rtc_tamper_disable.....	619
Table 3-922. Function rtc_output_pin_select.....	619
Table 3-923. Function rtc_alarm_output_config.....	620
Table 3-924. Function rtc_calibration_output_config.....	620
Table 3-925. Function rtc_hour_adjust.....	621
Table 3-926. Function rtc_second_adjust.....	622
Table 3-927. Function rtc_bypass_shadow_enable.....	622
Table 3-928. Function rtc_bypass_shadow_disable.....	623
Table 3-929. Function rtc_refclock_detection_enable.....	623
Table 3-930. Function rtc_refclock_detection_disable.....	624
Table 3-931. Function rtc_wakeup_enable.....	624
Table 3-932. Function rtc_wakeup_disable.....	625
Table 3-933. Function rtc_wakeup_clock_set.....	625
Table 3-934. Function rtc_wakeup_timer_set.....	626
Table 3-935. Function rtc_wakeup_timer_get.....	626
Table 3-936. Function rtc_smooth_calibration_config.....	627
Table 3-937. Function rtc_interrupt_enable.....	628
Table 3-938. Function rtc_interrupt_disable.....	628
Table 3-939. Function rtc_flag_get.....	629
Table 3-940. Function rtc_flag_clear.....	630
Table 3-941. HRTIMER Register.....	631
Table 3-942. HRTIMER firmware function.....	634
Table 3-943. Structure hrtimer_baseinit_parameter_struct.....	637
Table 3-944. Structure hrtimer_timerinit_parameter_struct.....	637
Table 3-945. Structure hrtimer_timercfg_parameter_struct.....	638
Table 3-946. Structure hrtimer_capture_value_struct.....	638
Table 3-947. Structure hrtimer_comparecfg_parameter_struct.....	638
Table 3-948. Structure hrtimer_exevcfg_parameter_struct.....	639
Table 3-949. Structure hrtimer_deadtimecfg_parameter_struct.....	639
Table 3-950. Structure hrtimer_carriersignalcfg_parameter_struct.....	639
Table 3-951. Structure hrtimer_synccfg_parameter_struct.....	640
Table 3-952. Structure hrtimer_bunchmode_parameter_struct.....	640
Table 3-953. Structure hrtimer_exeventcfg_parameter_struct.....	640
Table 3-954. Structure hrtimer_exeventcnt_parameter_struct.....	640
Table 3-955. Structure hrtimer_faultcfg_parameter_struct.....	641
Table 3-956. Structure hrtimer_adctrigcfg_parameter_struct.....	641
Table 3-957. Structure hrtimer_channel_outputcfg_parameter_struct.....	641
Table 3-958. Structure hrtimer_roll_over_parameter_struct.....	642
Table 3-959. Structure hrtimer_double_trigger_parameter_struct.....	642
Table 3-960. Function hrtimer_deinit.....	642
Table 3-961. Function hrtimer_dll_calibration_start.....	643
Table 3-962. Function hrtimer_baseinit_struct_para_init.....	643



Table 3-963. Function <code>hrtimer_timers_base_init</code> .....	644
Table 3-964. Function <code>hrtimer_timers_counter_enable</code> .....	645
Table 3-965. Function <code>hrtimer_timers_counter_enable</code> .....	646
Table 3-966. Function <code>hrtimer_timers_update_event_enable</code> .....	646
Table 3-967. Function <code>hrtimer_timers_update_event_enable</code> .....	647
Table 3-968. Function <code>hrtimer_software_update</code> .....	648
Table 3-969. Function <code>hrtimer_software_counter_reset</code> .....	648
Table 3-970. Function <code>hrtimer_output_exchange</code> .....	649
Table 3-971. Function <code>hrtimer_timerinit_struct_para_init</code> .....	650
Table 3-972. Function <code>hrtimer_timers_waveform_init</code> .....	650
Table 3-973. Function <code>hrtimer_timercfg_struct_para_init</code> .....	651
Table 3-974. Function <code>hrtimer_slavetimer_waveform_config</code> .....	652
Table 3-975. Function <code>hrtimer_comparecfg_struct_para_init</code> .....	653
Table 3-976. Function <code>hrtimer_slavetimer_waveform_compare_config</code> .....	654
Table 3-977. Function <code>hrtimer_channel_outputcfg_struct_para_init</code> .....	655
Table 3-978. Function <code>hrtimer_slavetimer_waveform_channel_config</code> .....	655
Table 3-979. Function <code>hrtimer_slavetimer_waveform_channel_software_request</code> .....	656
Table 3-980. Function <code>hrtimer_slavetimer_waveform_channel_output_level_get</code> .....	657
Table 3-981. Function <code>hrtimer_slavetimer_waveform_channel_state_get</code> .....	658
Table 3-982. Function <code>hrtimer_channel_outputcfg_struct_para_init</code> .....	659
Table 3-983. Function <code>hrtimer_slavetimer_waveform_channel_software_request</code> .....	660
Table 3-984. Function <code>hrtimer_channel_outputcfg_struct_para_init</code> .....	661
Table 3-985. Function <code>hrtimer_slavetimer_carriersignal_config</code> .....	661
Table 3-986. Function <code>hrtimer_output_channel_enable</code> .....	662
Table 3-987. Function <code>hrtimer_output_channel_disable</code> .....	663
Table 3-988. Function <code>hrtimer_slavetimer_waveform_compare_config</code> .....	663
Table 3-989. Function <code>hrtimer_slavetimer_compare_value_get</code> .....	664
Table 3-990. Function <code>hrtimer_mastertimer_compare_value_config</code> .....	665
Table 3-991. Function <code>hrtimer_slavetimer_compare_value_get</code> .....	666
Table 3-992. Function <code>hrtimer_timers_counter_value_config</code> .....	666
Table 3-993. Function <code>hrtimer_timers_counter_value_get</code> .....	667
Table 3-994. Function <code>hrtimer_timers_autoreload_value_config</code> .....	668
Table 3-995. Function <code>hrtimer_timers_autoreload_value_get</code> .....	668
Table 3-996. Function <code>hrtimer_timers_repetition_value_config</code> .....	669
Table 3-997. Function <code>hrtimer_timers_repetition_value_get</code> .....	670
Table 3-998. Function <code>hrtimer_exefilter_struct_para_init</code> .....	671
Table 3-999. Function <code>hrtimer_slavetimer_exeevent_filtering_config</code> .....	671
Table 3-1000. Function <code>hrtimer_exeeventcfg_struct_para_init</code> .....	672
Table 3-1001. Function <code>hrtimer_exeevent_config</code> .....	673
Table 3-1002. Function <code>hrtimer_exeevent_prescaler</code> .....	674
Table 3-1003. Function <code>hrtimer_exeevent_config</code> .....	674
Table 3-1004. Function <code>hrtimer_exeeventx_counter_config</code> .....	675
Table 3-1005. Function <code>hrtimer_software_reset_exeeventx_counter</code> .....	676
Table 3-1006. Function <code>hrtimer_exeeventx_counter_enable</code> .....	676

Table 3-1007. Function <code>hrtimer_exeventx_counter_disable</code> .....	677
Table 3-1008. Function <code>hrtimer_exeventx_counter_read</code> .....	678
Table 3-1009. Function <code>hrtimer_synccfg_struct_para_init</code> .....	678
Table 3-1010. Function <code>hrtimer_synchronization_config</code> .....	679
Table 3-1011. Function <code>hrtimer_double_channel_struct_para_init</code> .....	680
Table 3-1012. Function <code>hrtimer_double_trigger_config</code> .....	680
Table 3-1013. Function <code>hrtimer_roll_over_struct_para_init</code> .....	681
Table 3-1014. Function <code>hrtimer_roll_over_mode_config</code> .....	681
Table 3-1015. Function <code>hrtimer_faultcfg_struct_para_init</code> .....	682
Table 3-1016. Function <code>hrtimer_fault_config</code> .....	683
Table 3-1017. Function <code>hrtimer_fault_prescaler_config</code> .....	684
Table 3-1018. Function <code>hrtimer_fault_input_enable</code> .....	685
Table 3-1019. Function <code>hrtimer_fault_input_disable</code> .....	685
Table 3-1020. Function <code>hrtimer_fault_counter_reset</code> .....	686
Table 3-1021. Function <code>hrtimer_fault_blank_enable</code> .....	686
Table 3-1022. Function <code>hrtimer_fault_blank_disable</code> .....	687
Table 3-1023. Function <code>hrtimer_timers_dma_enable</code> .....	688
Table 3-1024. Function <code>hrtimer_timers_dma_enable</code> .....	689
Table 3-1025. Function <code>hrtimer_dmamode_config</code> .....	690
Table 3-1026. Function <code>hrtimer_bunchmode_struct_para_init</code> .....	692
Table 3-1027. Function <code>hrtimer_bunchmode_config</code> .....	692
Table 3-1028. Function <code>hrtimer_bunchmode_enable</code> .....	693
Table 3-1029. Function <code>hrtimer_bunchmode_disable</code> .....	694
Table 3-1030. Function <code>hrtimer_bunchmode_flag_get</code> .....	694
Table 3-1031. Function <code>hrtimer_bunchmode_software_start</code> .....	695
Table 3-1032. Function <code>hrtimer_slavetimer_capture_config</code> .....	695
Table 3-1033. Function <code>hrtimer_slavetimer_capture_software</code> .....	696
Table 3-1034. Function <code>hrtimer_slavetimer_capture_value_read</code> .....	697
Table 3-1035. Function <code>hrtimer_adctrigcfg_struct_para_init</code> .....	698
Table 3-1036. Function <code>hrtimer_adc_trigger0_3_config</code> .....	699
Table 3-1037. Function <code>hrtimer_adc_trigger_config</code> .....	700
Table 3-1038. Function <code>hrtimer_adc_prescaler_config</code> .....	700
Table 3-1039. Function <code>hrtimer_slavetimer_counter_direction_get</code> .....	701
Table 3-1040. Function <code>hrtimer_timers_flag_get</code> .....	702
Table 3-1041. Function <code>hrtimer_timers_flag_clear</code> .....	703
Table 3-1042. Function <code>hrtimer_common_flag_get</code> .....	704
Table 3-1043. Function <code>hrtimer_common_flag_clear</code> .....	705
Table 3-1044. Function <code>hrtimer_timers_interrupt_enable</code> .....	706
Table 3-1045. Function <code>hrtimer_timers_interrupt_disable</code> .....	707
Table 3-1046. Function <code>hrtimer_timers_interrupt_flag_get</code> .....	708
Table 3-1047. Function <code>hrtimer_timers_interrupt_flag_clear</code> .....	710
Table 3-1048. Function <code>hrtimer_common_interrupt_enable</code> .....	711
Table 3-1049. Function <code>hrtimer_common_interrupt_disable</code> .....	711
Table 3-1050. Function <code>hrtimer_common_interrupt_flag_get</code> .....	712

Table 3-1051. Function <code>hrtimer_common_interrupt_flag_clear</code> .....	713
Table 3-1052. SPI Registers .....	714
Table 3-1053. SPI firmware function .....	714
Table 3-1054. <code>spi_parameter_struct</code> .....	715
Table 3-1055. Function <code>spi_deinit</code> .....	716
Table 3-1056. Function <code>spi_struct_para_init</code> .....	716
Table 3-1057. Function <code>spi_init</code> .....	717
Table 3-1058. Function <code>spi_enable</code> .....	718
Table 3-1059. Function <code>spi_disable</code> .....	719
Table 3-1060. Function <code>spi_nss_output_enable</code> .....	719
Table 3-1061. Function <code>spi_nss_output_disable</code> .....	720
Table 3-1062. Function <code>spi_nss_internal_high</code> .....	720
Table 3-1063. Function <code>spi_nss_internal_low</code> .....	721
Table 3-1064. Function <code>spi_dma_enable</code> .....	721
Table 3-1065. Function <code>spi_dma_disable</code> .....	722
Table 3-1066. Function <code>spi_transmit_odd_config</code> .....	722
Table 3-1067. Function <code>spi_receive_odd_config</code> .....	723
Table 3-1068. Function <code>spi_data_frame_size_config</code> .....	724
Table 3-1069. Function <code>spi_fifo_access_size_config</code> .....	724
Table 3-1070. Function <code>spi_bidirectional_transfer_config</code> .....	725
Table 3-1071. Function <code>spi_data_transmit</code> .....	725
Table 3-1072. Function <code>spi_data_receive</code> .....	726
Table 3-1073. Function <code>spi_crc_polynomial_set</code> .....	726
Table 3-1074. Function <code>spi_crc_polynomial_get</code> .....	727
Table 3-1075. Function <code>spi_crc_length_set</code> .....	728
Table 3-1076. Function <code>spi_crc_on</code> .....	728
Table 3-1077. Function <code>spi_crc_off</code> .....	729
Table 3-1078. Function <code>spi_crc_next</code> .....	729
Table 3-1079. Function <code>spi_crc_get</code> .....	730
Table 3-1080. Function <code>spi_crc_error_clear</code> .....	730
Table 3-1081. Function <code>spi_ti_mode_enable</code> .....	731
Table 3-1082. Function <code>spi_ti_mode_disable</code> .....	731
Table 3-1083. Function <code>spi_nssp_mode_enable</code> .....	732
Table 3-1084. Function <code>spi_nssp_mode_disable</code> .....	732
Table 3-1085. Function <code>spi_quad_enable</code> .....	733
Table 3-1086. Function <code>spi_quad_disable</code> .....	733
Table 3-1087. Function <code>spi_quad_write_enable</code> .....	734
Table 3-1088. Function <code>spi_quad_read_enable</code> .....	734
Table 3-1089. Function <code>spi_quad_io23_output_enable</code> .....	735
Table 3-1090. Function <code>spi_quad_io23_output_disable</code> .....	735
Table 3-1091. Function <code>spi_format_error_clear</code> .....	736
Table 3-1092. Function <code>spi_i2s_flag_get</code> .....	737
Table 3-1093. Function <code>spi_interrupt_enable</code> .....	738
Table 3-1094. Function <code>spi_interrupt_disable</code> .....	738

Table 3-1095. Function spi_interrupt_flag_get .....	739
Table 3-1096. SYSCFG Registers .....	740
Table 3-1097. SYSCFG firmware function .....	741
Table 3-1098. Enum syscfg_interrupt_enum .....	742
Table 3-1099. Enum syscfg_flag_enum .....	742
Table 3-1100. Enum syscfg_sram_serrbits_enum .....	743
Table 3-1101. Enum syscfg_sram_erraddr_enum .....	743
Table 3-1102. Enum timer_channel_input_enum .....	743
Table 3-1103. Function syscfg_deinit .....	752
Table 3-1104. Function syscfg_i2c_fast_mode_plus_enable .....	752
Table 3-1105. Function syscfg_i2c_fast_mode_plus_disable .....	753
Table 3-1106. Function syscfg_exti_line_config .....	754
Table 3-1107. Function syscfg_pin_reset_mode_config .....	754
Table 3-1108. Function syscfg_trigsel_cla_reset_mode_config .....	755
Table 3-1109. Function syscfg_lockup_enable .....	755
Table 3-1110. Function syscfg_timer_input_source_select .....	756
Table 3-1111. Function syscfg_flash_bank_remap_set .....	757
Table 3-1112. Function syscfg_bootmode_memmap_select .....	757
Table 3-1113. Function syscfg_sram_ecc_single_correctable_bit_get .....	758
Table 3-1114. Function syscfg_sram_ecc_error_address_get .....	759
Table 3-1115. Function syscfg_tcmsram_erase .....	759
Table 3-1116. Function syscfg_tcmsram_erase_lock .....	760
Table 3-1117. Function syscfg_tcmsram_erase_unlock .....	760
Table 3-1118. Function syscfg_tcmsram_page_wp_enable .....	761
Table 3-1119. Function syscfg_io_compensation_config .....	761
Table 3-1120. Function syscfg_fpu_interrupt_enable .....	762
Table 3-1121. Function syscfg_fpu_interrupt_disable .....	762
Table 3-1122. Function syscfg_interrupt_flag_get .....	763
Table 3-1123. Function syscfg_interrupt_flag_clear .....	764
Table 3-1124. Function syscfg_interrupt_enable .....	764
Table 3-1125. Function syscfg_interrupt_disable .....	765
Table 3-1126. Function syscfg_tcmsram_busy_flag_get .....	765
Table 3-1127. Function syscfg_compensation_cell_ready_flag_get .....	766
Table 3-1128. TIMERx Registers .....	766
Table 3-1129. TIMERx firmware function .....	768
Table 3-1130. Structure timer_parameter_struct .....	772
Table 3-1131. Structure timer_break_parameter_struct .....	772
Table 3-1132. Structure timer_oc_parameter_struct .....	773
Table 3-1133. Structure timer_omc_parameter_struct .....	774
Table 3-1134. Structure timer_ic_parameter_struct .....	774
Table 3-1135. Structure timer_free_complementary_parameter_struct .....	774
Table 3-1136. Function timer_deinit .....	774
Table 3-1137. Function timer_struct_para_init .....	775
Table 3-1138. Function timer_init .....	776

Table 3-1139. Function timer_enable .....	776
Table 3-1140. Function timer_disable .....	777
Table 3-1141. Function timer_auto_reload_shadow_enable .....	778
Table 3-1142. Function timer_auto_reload_shadow_disable .....	778
Table 3-1143. Function timer_update_event_enable.....	779
Table 3-1144. Function timer_update_event_disable.....	779
Table 3-1145. Function timer_counter_alignment .....	780
Table 3-1146. Function timer_counter_up_direction .....	780
Table 3-1147. Function timer_counter_down_direction .....	781
Table 3-1148. Function timer_adjustment_mode_config .....	781
Table 3-1149. Function timer_prescaler_config.....	782
Table 3-1150. Function timer_repetition_value_config.....	783
Table 3-1151. Function timer_runtime_repetition_value_read .....	784
Table 3-1152. Function timer_autoreload_value_config .....	784
Table 3-1153. Function timer_auto_reload_fract_value_config .....	785
Table 3-1154. Function timer_autoreload_value_read .....	785
Table 3-1155. Function timer_auto_reload_fract_value_read .....	786
Table 3-1156. Function timer_counter_value_config .....	787
Table 3-1157. Function timer_counter_read .....	787
Table 3-1158. Function timer_prescaler_read .....	788
Table 3-1159. Function timer_single_pulse_mode_config .....	788
Table 3-1160. Function timer_delayable_single_pulse_mode_config .....	789
Table 3-1161. Function timer_update_source_config .....	790
Table 3-1162. Function timer_ocpre_clear_source_config .....	791
Table 3-1163. Function timer_ocpre_clear_input_config .....	792
Table 3-1164. Function timer_pulse_on_compare_config .....	792
Table 3-1165. Function timer_dma_enable .....	793
Table 3-1166. Function timer_dma_disable .....	794
Table 3-1167. Function timer_channel_dma_request_source_select.....	795
Table 3-1168. Function timer_dma_transfer_config.....	795
Table 3-1169. Function timer_event_software_generate.....	798
Table 3-1170. Function timer_break_struct_para_init .....	799
Table 3-1171. Function timer_break_config.....	800
Table 3-1172. Function timer_break_enable .....	801
Table 3-1173. Function timer_break_disable .....	802
Table 3-1174. Function timer_automatic_output_enable .....	802
Table 3-1175. Function timer_automatic_output_disable .....	803
Table 3-1176. Function timer_primary_output_config .....	803
Table 3-1177. Function timer_channel_control_shadow_config .....	804
Table 3-1178. Function timer_channel_control_shadow_update_config.....	805
Table 3-1179. Function timer_channel_output_struct_para_init .....	806
Table 3-1180. Function timer_channel_output_config .....	806
Table 3-1181. Function timer_channel_output_mode_config .....	807
Table 3-1182. Function timer_channel_output_pulse_value_config.....	809

Table 3-1183. Function timer_channel_output_pulse_fract_value_config .....	810
Table 3-1184. Function timer_channel_output_shadow_config .....	810
Table 3-1185. Function timer_channel_output_clear_config .....	811
Table 3-1186. Function timer_channel_output_compare_fast_config .....	812
Table 3-1187. Function timer_channel_output_polarity_config .....	813
Table 3-1188. Function timer_channel_complementary_output_polarity_config .....	815
Table 3-1189. Function timer_channel_output_state_config .....	815
Table 3-1190. Function timer_channel_complementary_output_state_config .....	816
Table 3-1191. Function timer_channel_input_struct_para_init .....	817
Table 3-1192. Function timer_input_capture_config .....	818
Table 3-1193. Function timer_channel_input_capture_prescaler_config .....	819
Table 3-1194. Function timer_channel_capture_value_register_read .....	820
Table 3-1195. Function timer_input_pwm_capture_config .....	821
Table 3-1196. Function timer_hall_mode_config .....	821
Table 3-1197. Function timer_multi_mode_channel_output_parameter_struct_init .....	822
Table 3-1198. Function timer_multi_mode_channel_output_config .....	823
Table 3-1199. Function timer_multi_mode_channel_mode_config .....	824
Table 3-1200. Function timer_input_trigger_source_select .....	824
Table 3-1201. Function timer_master_output0_trigger_source_select .....	826
Table 3-1202. Function timer_master_output1_trigger_source_select .....	827
Table 3-1203. Function timer_slave_mode_select .....	828
Table 3-1204. Function timer_pause_reset_polarity_config .....	829
Table 3-1205. Function timer_master_slave_mode_config .....	830
Table 3-1206. Function timer_external_trigger_config .....	830
Table 3-1207. Function timer_quadrature_decoder_mode_config .....	831
Table 3-1208. Function timer_decoder_mode_config .....	833
Table 3-1209. Function timer_internal_clock_config .....	834
Table 3-1210. Function timer_internal_trigger_as_external_clock_config .....	834
Table 3-1211. Function timer_external_trigger_as_external_clock_config .....	835
Table 3-1212. Function timer_external_clock_mode0_config .....	836
Table 3-1213. Function timer_external_clock_mode1_config .....	837
Table 3-1214. Function timer_external_clock_mode1_disable .....	838
Table 3-1215. Function timer_write_chxval_register_config .....	839
Table 3-1216. Function timer_output_value_selection_config .....	839
Table 3-1217. Function timer_commutation_control_shadow_register_config .....	840
Table 3-1218. Function timer_output_match_pulse_select .....	841
Table 3-1219. Function timer_channel_composite_pwm_mode_config .....	842
Table 3-1220. Function timer_channel_composite_pwm_mode_output_pulse_value_config .....	842
Table 3-1221. Function timer_channel_additional_compare_value_config .....	843
Table 3-1222. Function timer_channel_additional_output_shadow_config .....	844
Table 3-1223. Function timer_channel_additional_output_update_select .....	845
Table 3-1224. Function timer_channel_additional_compare_value_read .....	846
Table 3-1225. Function timer_break_external_source_config .....	846
Table 3-1226. Function timer_break_external_polarity_config .....	847



Table 3-1227. Function timer_break_lock_config.....	849
Table 3-1228. Function timer_break_lock_release_config .....	849
Table 3-1229. Function timer_dead_time_falling_edge_config.....	850
Table 3-1230. Function timer_dead_time_different_config .....	851
Table 3-1231. Function timer_dead_time_modify_config .....	851
Table 3-1232. Function timer_channel_break_control_config .....	852
Table 3-1233. Function timer_channel_dead_time_config .....	853
Table 3-1234. Function timer_free_complementary_struct_para_init .....	853
Table 3-1235. Function timer_channel_free_complementary_config .....	854
Table 3-1236. Function timer_watchdog_value_config .....	855
Table 3-1237. Function timer_watchdog_value_read .....	856
Table 3-1238. Function timer_decoder_disconnection_detection_config .....	856
Table 3-1239. Function timer_decoder_jump_detection_config .....	857
Table 3-1240. Function timer_decoder_modify_config .....	857
Table 3-1241. Function timer_decoder_mode_update_source_config .....	858
Table 3-1242. Function timer_index_reset_counter_config .....	859
Table 3-1243. Function timer_index_reset_direction_config .....	859
Table 3-1244. Function timer_decoder_index_position_config .....	860
Table 3-1245. Function timer_first_index_reset_counter_config .....	861
Table 3-1246. Function timer_upif_backup_config .....	861
Table 3-1247. Function timer_upifbu_bit_get.....	862
Table 3-1248. Function timer_counter_initial_register_config.....	863
Table 3-1249. Function timer_counter_initial_config.....	863
Table 3-1250. Function timer_synchronization_event_generate .....	864
Table 3-1251. Function timer_flag_get .....	865
Table 3-1252. Function timer_flag_clear .....	866
Table 3-1253. Function timer_interrupt_enable .....	868
Table 3-1254. Function timer_interrupt_disable .....	869
Table 3-1255. Function timer_interrupt_flag_get.....	870
Table 3-1256. Function timer_interrupt_flag_clear.....	872
Table 3-1257. TMU Registers .....	873
Table 3-1258. TMU firmware function .....	873
Table 3-1259. Structure tmu_parameter_struct .....	874
Table 3-1260. Function tmu_deinit .....	875
Table 3-1261. Function tmu_struct_para_init.....	875
Table 3-1262. Function tmu_init.....	876
Table 3-1263. Function tmu_dma_read_enable .....	877
Table 3-1264. Function tmu_dma_read_disable .....	877
Table 3-1265. Function tmu_dma_write_enable .....	878
Table 3-1266. Function tmu_dma_write_disable .....	878
Table 3-1267. Function tmu_one_q31_write .....	879
Table 3-1268. Function tmu_two_q31_write .....	879
Table 3-1269. Function tmu_two_q15_write .....	880
Table 3-1270. Function tmu_one_f32_write .....	880

Table 3-1271. Function tmu_two_f32_write .....	881
Table 3-1272. Function tmu_one_q31_read .....	881
Table 3-1273. Function tmu_two_q31_read .....	882
Table 3-1274. Function tmu_two_q15_read .....	883
Table 3-1275. Function tmu_one_f32_read .....	883
Table 3-1276. Function tmu_two_f32_read .....	884
Table 3-1277. Function tmu_flag_get .....	884
Table 3-1278. Function tmu_flag_clear .....	885
Table 3-1279. Function tmu_interrupt_enable .....	885
Table 3-1280. Function tmu_interrupt_disable .....	886
Table 3-1281. Function tmu_interrupt_flag_get .....	886
Table 3-1282. Function tmu_interrupt_flag_clear .....	887
Table 3-1283. TRIGSEL Registers .....	888
Table 3-1284. TRIGSEL firmware function .....	889
Table 3-1285. Enum trigsel_source_enum .....	889
Table 3-1286. Enum trigsel_periph_enum .....	894
Table 3-1287. Function trigsel_deinit .....	896
Table 3-1288. Function trigsel_init .....	896
Table 3-1289. Function trigsel_trigger_source_get .....	897
Table 3-1290. Function trigsel_register_lock_set .....	897
Table 3-1291. Function trigsel_register_lock_get .....	898
Table 3-1292. TRNG Registers .....	899
Table 3-1293. TRNG firmware function .....	899
Table 3-1294. Enum trng_inmod_enum .....	900
Table 3-1295. Enum trng_outmod_enum .....	900
Table 3-1296. Enum trng_modsel_enum .....	900
Table 3-1297. Enum trng_nist_seed_enum .....	900
Table 3-1298. Enum trng_flag_enum .....	900
Table 3-1299. Enum trng_int_flag_enum .....	901
Table 3-1300. Function trng_deinit .....	901
Table 3-1301. Function trng_enable .....	901
Table 3-1302. Function trng_disable .....	902
Table 3-1303. Function trng_lock .....	902
Table 3-1304. Function trng_mode_config .....	903
Table 3-1305. Function trng_nist_seed_config .....	904
Table 3-1306. Function trng_postprocessing_enable .....	904
Table 3-1307. Function trng_postprocessing_disable .....	905
Table 3-1308. Function trng_conditioning_enable .....	906
Table 3-1309. Function trng_conditioning_disable .....	907
Table 3-1310. Function trng_disable .....	908
Table 3-1311. Function trng_conditioning_output_bitwidth .....	908
Table 3-1312. Function trng_replace_test_enable .....	909
Table 3-1313. Function trng_replace_test_disable .....	910
Table 3-1314. Function trng_hash_init_enable .....	910



Table 3-1315. Function <code>trng_hash_init_disable</code> .....	911
Table 3-1316. Function <code>trng_powermode_config</code> .....	912
Table 3-1317. Function <code>trng_disable</code> .....	912
Table 3-1318. Function <code>trng_clockerror_detection_enable</code> .....	913
Table 3-1319. Function <code>trng_clockerror_detection_disable</code> .....	914
Table 3-1320. Function <code>trng_get_true_random_data</code> .....	914
Table 3-1321. Function <code>trng_conditioning_reset_enable</code> .....	915
Table 3-1322. Function <code>trng_conditioning_reset_disable</code> .....	916
Table 3-1323. Function <code>trng_conditioning_algo_config</code> .....	916
Table 3-1324. Function <code>trng_health_tests_config</code> .....	917
Table 3-1325. Function <code>trng_flag_get</code> .....	918
Table 3-1326. Function <code>trng_interrupt_enable</code> .....	919
Table 3-1327. Function <code>trng_interrupt_disable</code> .....	919
Table 3-1328. Function <code>trng_interrupt_flag_get</code> .....	920
Table 3-1329. Function <code>trng_interrupt_flag_clear</code> .....	920
Table 3-1330. USART Registers .....	921
Table 3-1331. USART firmware function .....	921
Table 3-1332. Enum <code>usart_flag_enum</code> .....	924
Table 3-1333. Enum <code>usart_interrupt_flag_enum</code> .....	925
Table 3-1334. Enum <code>usart_interrupt_enum</code> .....	925
Table 3-1335. Enum <code>usart_invert_enum</code> .....	926
Table 3-1336. Function <code>usart_deinit</code> .....	926
Table 3-1337. Function <code>usart_baudrate_set</code> .....	927
Table 3-1338. Function <code>usart_parity_config</code> .....	927
Table 3-1339. Function <code>usart_word_length_set</code> .....	928
Table 3-1340. Function <code>usart_stop_bit_set</code> .....	929
Table 3-1341. Function <code>usart_enable</code> .....	929
Table 3-1342. Function <code>usart_disable</code> .....	930
Table 3-1343. Function <code>usart_transmit_config</code> .....	930
Table 3-1344. Function <code>usart_receive_config</code> .....	931
Table 3-1345. Function <code>usart_data_first_config</code> .....	932
Table 3-1346. Function <code>usart_invert_config</code> .....	932
Table 3-1347. Function <code>usart_overrun_enable</code> .....	933
Table 3-1348. Function <code>usart_overrun_disable</code> .....	933
Table 3-1349. Function <code>usart_oversample_config</code> .....	934
Table 3-1350. Function <code>usart_sample_bit_config</code> .....	935
Table 3-1351. Function <code>usart_receiver_timeout_enable</code> .....	935
Table 3-1352. Function <code>usart_receiver_timeout_disable</code> .....	936
Table 3-1353. Function <code>usart_receiver_timeout_threshold_config</code> .....	936
Table 3-1354. Function <code>usart_data_transmit</code> .....	937
Table 3-1355. Function <code>usart_data_receive</code> .....	937
Table 3-1356. Function <code>usart_command_enable</code> .....	938
Table 3-1357. Function <code>usart_address_0_match_mode_enable</code> .....	939
Table 3-1358. Function <code>usart_address_0_match_mode_disable</code> .....	939

Table 3-1359. Function usart_address_1_match_mode_enable.....	940
Table 3-1360. Function usart_address_1_match_mode_disable.....	940
Table 3-1361. Function usart_address_0_config.....	941
Table 3-1362. Function usart_address_1_config.....	941
Table 3-1363. Function usart_address_0_detection_mode_config.....	942
Table 3-1364. Function usart_address_1_detection_mode_config.....	943
Table 3-1365. Function usart_mute_mode_enable.....	943
Table 3-1366. Function usart_mute_mode_disable .....	944
Table 3-1367. Function usart_mute_mode_wakeup_config .....	944
Table 3-1368. Function usart_lin_mode_enable .....	945
Table 3-1369. Function usart_lin_mode_disable .....	946
Table 3-1370. Function usart_lin_break_dection_length_config.....	946
Table 3-1371. Function usart_halfduplex_enable.....	947
Table 3-1372. Function usart_halfduplex_disable .....	947
Table 3-1373. Function usart_clock_enable .....	948
Table 3-1374. Function usart_clock_disable .....	948
Table 3-1375. Function usart_synchronous_clock_config .....	949
Table 3-1376. Function usart_guard_time_config .....	950
Table 3-1377. Function usart_smartcard_mode_enable .....	950
Table 3-1378. Function usart_smartcard_mode_disable .....	951
Table 3-1379. Function usart_smartcard_mode_nack_enable.....	951
Table 3-1380. Function usart_smartcard_mode_nack_disable .....	952
Table 3-1381. Function usart_smartcard_mode_early_nack_enable.....	952
Table 3-1382. Function usart_smartcard_mode_early_nack_disable.....	953
Table 3-1383. Function usart_smartcard_autoretry_config.....	953
Table 3-1384. Function usart_block_length_config .....	954
Table 3-1385. Function usart_irda_mode_enable.....	954
Table 3-1386. Function usart_irda_mode_disable.....	955
Table 3-1387. Function usart_prescaler_config.....	955
Table 3-1388. Function usart_irda_lowpower_config .....	956
Table 3-1389. Function usart_hardware_flow_rts_config .....	956
Table 3-1390. Function usart_hardware_flow_cts_config .....	957
Table 3-1391. Function usart_hardware_flow_coherence_config .....	958
Table 3-1392. Function usart_rs485_driver_enable .....	958
Table 3-1393. Function usart_rs485_driver_disable .....	959
Table 3-1394. Function usart_driver_assertime_config .....	960
Table 3-1395. Function usart_driver_deassertime_config.....	960
Table 3-1396. Function usart_depolarity_config .....	961
Table 3-1397. Function usart_dma_receive_config .....	961
Table 3-1398. Function usart_dma_transmit_config.....	962
Table 3-1399. Function usart_reception_error_dma_disable.....	963
Table 3-1400. Function usart_reception_error_dma_enable .....	963
Table 3-1401. Function usart_wakeup_enable .....	964
Table 3-1402. Function usart_wakeup_disable .....	964

Table 3-1403. Function usart_wakeup_mode_config .....	965
Table 3-1404. Function usart_fifo_enable .....	965
Table 3-1405. Function usart_fifo_disable .....	966
Table 3-1406. Function usart_transmit_fifo_threshold_config .....	967
Table 3-1407. Function usart_receive_fifo_threshold_config .....	967
Table 3-1408. Function usart_receive_fifo_counter_number .....	968
Table 3-1409. Function usart_flag_get .....	969
Table 3-1410. Function usart_flag_clear .....	970
Table 3-1411. Function usart_interrupt_enable .....	971
Table 3-1412. Function usart_interrupt_disable .....	971
Table 3-1413. Function usart_interrupt_flag_get .....	972
Table 3-1414. Function usart_interrupt_flag_clear .....	972
Table 3-1415. VREF Registers .....	974
Table 3-1416. VREF firmware function .....	974
Table 3-1417. Function vref_deinit .....	974
Table 3-1418. Function vref_enable .....	975
Table 3-1419. Function vref_disable .....	975
Table 3-1420. Function vref_high_impedance_mode_enable .....	976
Table 3-1421. Function vref_high_impedance_mode_disable .....	976
Table 3-1422. Function vref_status_get .....	977
Table 3-1423. Function vref_voltage_select .....	977
Table 3-1424. Function vref_calib_value_set .....	978
Table 3-1425. Function vref_calib_value_get .....	979
Table 3-1426. WWDGT Registers .....	979
Table 3-1427. WWDGT firmware function .....	979
Table 3-1428. Function wwdgt_deinit .....	980
Table 3-1429. Function wwdgt_enable .....	980
Table 3-1430. Function wwdgt_counter_update .....	981
Table 3-1431. Function wwdgt_config .....	981
Table 3-1432. Function wwdgt_interrupt_enable .....	982
Table 3-1433. Function wwdgt_flag_get .....	982
Table 3-1434. Function wwdgt_flag_clear .....	983
Table 4-1. Revision history .....	984

## 1. Introduction

This manual introduces firmware library of GD32G5x3 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 GD32G5x3 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
CAN	Controller area network
CAU	Cryptographic Acceleration Unit
CLA	Configurable Logic Array
CMP	Comparator

Peripherals	Descriptions
CRC	CRC calculation unit
DAC	Digital-to-analog converter
DBG	Debug
DMA	Direct memory access controller
DMAMUX	DMA request multiplexer
EXMC	External memory controller
EXTI	Interrupt/event controller
FAC	Filter arithmetic accelerator
FFT	Fast Fourier Transform
FMC	Flash memory controller
FWDGT	Free watchdog timer
GPIO	General-purpose I/Os
HPDF	High-Performance Digital Filter
I2C	Inter-integrated circuit interface
LPTIMER	Low power timer
MISC	Nested Vectored Interrupt Controller
PMU	Power management unit
QSPI	Quad-SPI interface
RCU	Reset and clock unit
RTC	Reset and clock unit
HRTIMER	High-Resolution Timer
SPI/I2S	Secure digital input/output interface
SYSCFG	System configuration
TIMER	TIMER
TMU	Trigonometric Math Unit
TRIGSEL	Trigger selection controller
TRNG	True random number generator
USART	Universal synchronous / asynchronous receiver / transmitter
VREF	VREF
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 “gd32g5x3\_”, such as: gd32g5x3\_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;

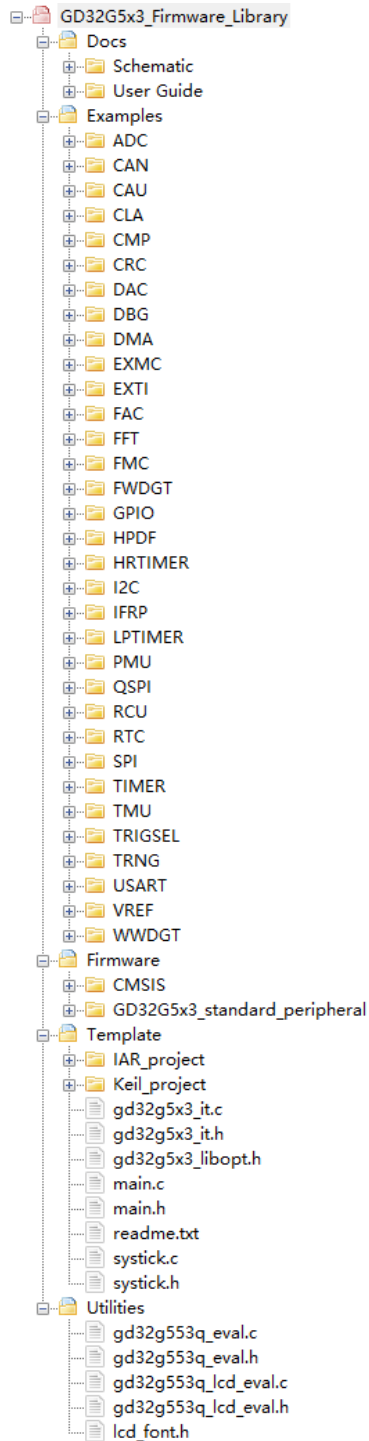
- Registers are handled as constants. The naming of them are written in uppercase of English letters. In most cases, register names are shortened accord with the user manual;
- Variables are written in lowercase, when concluded by several words, underlines should be adapted among words;
- The naming 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 lower case.

## 2. Firmware Library Overview

### 2.1. File Structure of Firmware Library

GD32G5x3\_Firmware\_Library, the file structure is shown as below:

**Figure 2-1. File structure of firmware library of GD32G5x3**





### 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;
- gd32g5x3\_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);
- gd32g5x3\_it.c: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- gd32g5x3\_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 GD32G5x3 and system configuration file;
- GD32G5x3\_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.

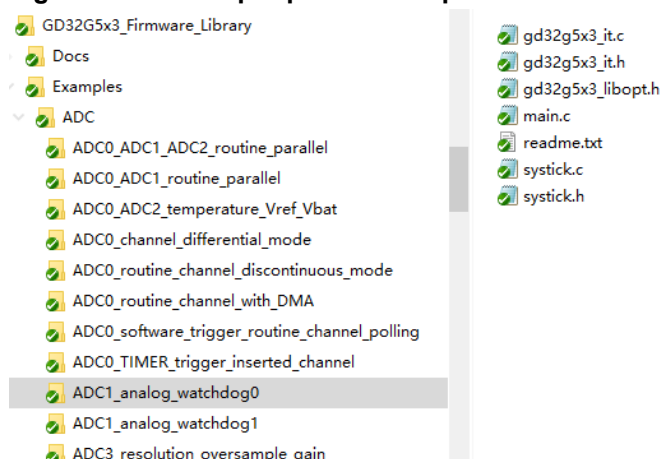
### 2.1.3. Template Folder

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 ADC, open “ADC” folder, select an example of ADC, such as “ADC1\_analog\_watchdog0”, shown as below:

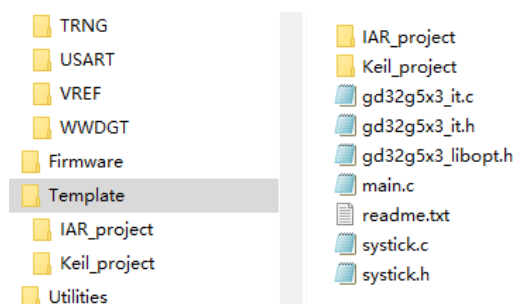
**Figure 2-2. Select peripheral example files**



## 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 “ADC1\_analog\_watchdog0” folder to the “Template” subfolder, shown as below:

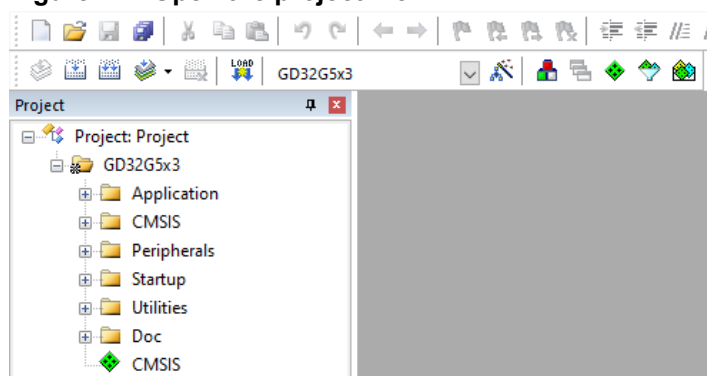
**Figure 2-3. Copy the peripheral example files**



## Open a project

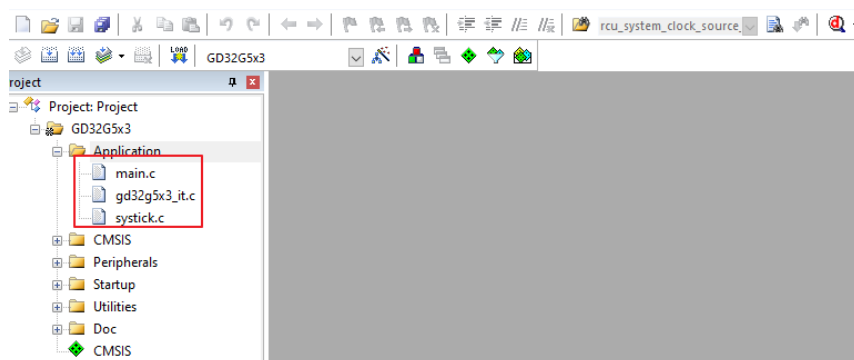
GD provides project in Keil and IAR, users can open project in different IDEs according to their need, such as “Keil\_project”, open \Template\Keil\_project\Project.uvprojx, shown as below:

**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:

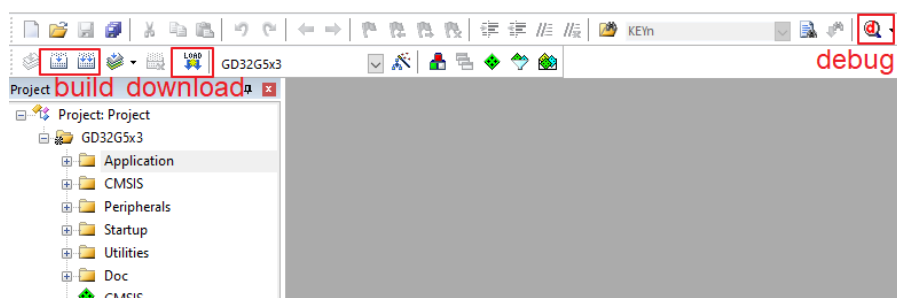
**Figure 2-5. Configure project files**



## 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:

- gd32g553q\_eval.h is related header file of the evaluation board about running the firmware examples;
- gd32g553q\_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
gd32g5x3_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.
gd32g5x3_it.h	Header file, including all the prototypes of interrupt service routines.
gd32g5x3_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.
gd32g5x3_xxx.h	The header file of peripheral xxx, including functions about peripheral xxx, and the variables used for functions.
gd32g5x3_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**

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

### 3.2. ADC

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**

<b>Registers</b>	<b>Descriptions</b>
ADC_STAT	ADC status register
ADC_CTL0	ADC control register 0
ADC_CTL1	ADC control register 1
ADC_IOFFx	ADC inserted channel data offset register x(x=0..3)
ADC_WDHT0	ADC watchdog high threshold register 0
ADC_WDLT0	ADC watchdog low threshold register 0
ADC_RSQx	ADC routine sequence register x(x=0..8)

Registers	Descriptions
ADC_ISQx	ADC inserted sequence register x(x=0..2)
ADC_IDATAx	ADC inserted data register x(x=0..3)
ADC_RDATA	ADC routine data register
ADC_OVSAMPCTL	ADC oversampling control register
ADC_WD1SR	ADC watchdog 1 channel selection register
ADC_WD2SR	ADC watchdog 2 channel selection register
ADC_WDHT1	ADC watchdog high threshold register 1
ADC_WDLT1	ADC watchdog low threshold register 1
ADC_WDHT2	ADC watchdog high threshold register 2
ADC_WDLT2	ADC watchdog low threshold register 2
ADC_DIFCTL	ADC differential mode control register
ADC_GAINCFG	ADC gain configure register
ADC_SSTAT	ADC summary status register
ADC_SYNCCTL	ADC sync control register
ADC_SYNCDATA	ADC sync routine data register

### 3.2.2. Descriptions of Peripheral functions

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

**Table 3-3. ADC firmware function**

Function name	Function description
adc_deinit	reset ADC
adc_clock_config	configure the ADC clock
adc_special_function_config	enable or disable ADC special function
adc_data_alignment_config	configure ADC data alignment
adc_enable	enable ADC interface
adc_disable	disable ADC interface
adc_calibration_mode_config	configure ADC calibration mode
adc_calibration_number	configure ADC calibration number
adc_calibration_enable	ADC calibration and reset calibration
adc_resolution_config	configure ADC resolution
adc_internal_channel_config	enable or disable ADC internal channels
adc_gain_mode_enable	enable ADC gain mode
adc_gain_mode_disable	disable ADC gain mode
adc_gain_factor_set	configure ADC gain factor
adc_dma_mode_enable	enable DMA request
adc_dma_mode_disable	disable DMA request
adc_dma_request_after_last_enable	when DMA=1, the DMA engine issues a request at end of each routine conversion
adc_dma_request_after_last_disable	the DMA engine is disabled after the end of transfer signal from DMA controller is detected

Function name	Function description
adc_hpdf_mode_enable	enable hpdf mode
adc_hpdf_mode_disable	disable hpdf mode
adc_discontinuous_mode_config	configure ADC discontinuous mode
adc_channel_length_config	configure the length of routine sequence or inserted sequence
adc_routine_channel_config	configure ADC routine channel
adc_inserted_channel_config	configure ADC inserted channel
adc_inserted_channel_offset_config	configure ADC inserted channel offset
adc_inserted_channel_mean_value_mode_enable	enable ADC mean value mode for inserted channel
adc_inserted_channel_mean_value_mode_disable	disable ADC mean value mode for inserted channel
adc_inserted_channel_mean_value_mode_config	configure ADC mean value mode for channel
adc_channel_differential_mode_config	configure differential mode for channel
adc_external_trigger_config	enable ADC external trigger
adc_software_trigger_enable	enable ADC software trigger
adc_end_of_conversion_config	configure end of conversion mode
adc_routine_data_read	read ADC routine sequence data register
adc_inserted_data_read	read ADC inserted sequence data register
adc_watchdog0_single_channel_enable	configure ADC analog watchdog 0 single channel
adc_watchdog0_sequence_channel_enable	configure ADC analog watchdog 0 sequence channel
adc_watchdog0_disable	disable ADC analog watchdog 0
adc_watchdog1_channel_config	configure ADC analog watchdog 1 channel
adc_watchdog2_channel_config	configure ADC analog watchdog 2 channel
adc_watchdog1_disable	disable ADC analog watchdog 1
adc_watchdog2_disable	disable ADC analog watchdog 2
adc_watchdog0_threshold_config	configure ADC analog watchdog 0 threshold
adc_watchdog1_threshold_config	configure ADC analog watchdog 1 threshold
adc_watchdog2_threshold_config	configure ADC analog watchdog 2 threshold
adc_oversample_mode_config	configure ADC oversample mode
adc_oversample_mode_enable	enable ADC oversample mode
adc_oversample_mode_disable	disable ADC oversample mode
adc_flag_get	get the ADC flag bits
adc_flag_clear	clear the ADC flag bits
adc_interrupt_enable	enable ADC interrupt
adc_interrupt_disable	disable ADC interrupt
adc_interrupt_flag_get	get the ADC interrupt bits



Function name	Function description
adc_interrupt_flag_clear	clear the ADC flag
adc_sync_mode_config	configure the ADC sync mode
adc_sync_delay_config	configure the delay between 2 sampling phases in ADC sync modes
adc_sync_dma_config	configure ADC sync DMA mode selection
adc_sync_dma_request_after_last_enable	configure ADC sync DMA engine issues requests according to the SYNC DMA bits
adc_sync_dma_request_after_last_disable	configure ADC sync DMA engine is disabled after the end of transfer signal from DMA controller is detected
adc_sync_master_adc_routine_data_read	read ADC sync master adc routine data register
adc_sync_slave_adc_routine_data_read	read ADC sync slave adc routine data register

### 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
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### adc\_clock\_config

The description of adc\_clock\_config is shown as below:

**Table 3-5. Function adc\_clock\_config**

Function name	adc_clock_config
Function prototype	void adc_clock_config(uint32_t adc_periph, uint32_t prescaler);

<b>Function descriptions</b>	configure the ADC clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>prescaler</b>	configure ADCs prescaler ratio
<i>ADC_CLK_SYNC_HCLK_DIV2</i>	ADC sync clock mode HCLK div2
<i>ADC_CLK_SYNC_HCLK_DIV4</i>	ADC sync clock mode HCLK div4
<i>ADC_CLK_SYNC_HCLK_DIV6</i>	ADC sync clock mode HCLK div6
<i>ADC_CLK_SYNC_HCLK_DIV8</i>	ADC sync clock mode HCLK div8
<i>ADC_CLK_SYNC_HCLK_DIV10</i>	ADC sync clock mode HCLK div10
<i>ADC_CLK_SYNC_HCLK_DIV12</i>	ADC sync clock mode HCLK div12
<i>ADC_CLK_SYNC_HCLK_DIV14</i>	ADC sync clock mode HCLK div14
<i>ADC_CLK_SYNC_HCLK_DIV16</i>	ADC sync clock mode HCLK div16
<i>ADC_CLK_ASYNC_DIV1</i>	ADC async clock mode div1
<i>ADC_CLK_ASYNC_DIV2</i>	ADC async clock mode div2
<i>ADC_CLK_ASYNC_DIV4</i>	ADC async clock mode div4
<i>ADC_CLK_ASYNC_DIV6</i>	ADC async clock mode div6
<i>ADC_CLK_ASYNC_DIV8</i>	ADC async clock mode div8
<i>ADC_CLK_ASYNC_DIV10</i>	ADC async clock mode div10
<i>ADC_CLK_ASYNC_DIV12</i>	ADC async clock mode div12
<i>ADC_CLK_ASYNC_DIV16</i>	ADC async clock mode div16
<i>ADC_CLK_ASYNC_DIV32</i>	ADC async clock mode div32
<i>ADC_CLK_ASYNC_DIV64</i>	ADC async clock mode div64

V64	
ADC_CLK_ASYNC_DIV128	ADC async clock mode div128
V256	
ADC_CLK_ASYNC_DIV256	ADC async clock mode div256
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the ADC0 clock: HCLK div2 */
```

```
adc_clock_config(ADC0, ADC_CLK_SYNC_HCLK_DIV2);
```

### adc\_special\_function\_config

The description of adc\_special\_function\_config is shown as below:

**Table 3-6. Function adc\_special\_function\_config**

<b>Function name</b>	adc_special_function_config
<b>Function prototype</b>	void adc_special_function_config(uint32_t adc_periph, uint32_t function, ControlStatus newvalue);
<b>Function descriptions</b>	enable or disable ADC special function
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
Input parameter{in}	
<b>function</b>	the function to config
ADC_SCAN_MODE	scan mode select
ADC_INSERTED_CHANNEL_AUTO	inserted sequence convert automatically
ADC_CONTINUOUS_MODE	continuous mode select
Input parameter{in}	
<b>newvalue</b>	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC0 scan mode */

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

### adc\_data\_alignment\_config

The description of adc\_data\_alignment\_config is shown as below:

**Table 3-7. Function adc\_data\_alignment\_config**

<b>Function name</b>	adc_data_alignment_config
<b>Function prototype</b>	void adc_data_alignment_config(uint32_t adc_periph, uint32_t data_alignment);
<b>Function descriptions</b>	configure ADCx data alignment
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>data_alignment</b>	data alignment select
ADC_DATAALIGN_RIGHT	LSB alignment
ADC_DATAALIGN_LEFT	MSB alignment
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 data alignment */

adc_data_alignment_config(ADC0, ADC_DATAALIGN_RIGHT);
```

### adc\_enable

The description of adc\_enable is shown as below:

**Table 3-8. Function adc\_enable**

<b>Function name</b>	adc_enable
<b>Function prototype</b>	void adc_enable(uint32_t adc_periph);
<b>Function descriptions</b>	enable ADC interface
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</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 3-9. Function adc\_disable**

<b>Function name</b>	adc_disable
<b>Function prototype</b>	void adc_disable(uint32_t adc_periph);
<b>Function descriptions</b>	disable ADC interface
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC0 */
adc_disable(ADC0);
```

### adc\_calibration\_mode\_config

The description of adc\_calibration\_mode\_config is shown as below:

**Table 3-10. Function adc\_calibration\_mode\_config**

<b>Function name</b>	adc_calibration_mode_config
<b>Function prototype</b>	void adc_calibration_mode_config(uint32_t adc_periph, uint32_t clb_mode);
<b>Function descriptions</b>	configure ADC calibration mode
<b>Precondition</b>	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>clb_mode</b>	calibration mode
<i>ADC_CALIBRATION_OFFSET_MISMATCH</i>	ADC calibration offset and mismatch mode
<i>ADC_CALIBRATION_OFFSET</i>	ADC calibration offset mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 calibration mode */
```

```
adc_calibration_mode_config (ADC0, ADC_CALIBRATION_OFFSET);
```

### adc\_calibration\_number

The description of adc\_calibration\_number is shown as below:

**Table 3-11. Function adc\_calibration\_number**

<b>Function name</b>	adc_calibration_number
<b>Function prototype</b>	void adc_calibration_number(uint32_t adc_periph, uint32_t clb_num);
<b>Function descriptions</b>	configure ADC calibration number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>clb_num</b>	calibration number
<i>ADC_CALIBRATION_NUMBER_1</i>	calibrate once
<i>ADC_CALIBRATION_NUMBER_2</i>	calibrate twice
<i>ADC_CALIBRATION_NUMBER_4</i>	calibrate 4 times
<i>ADC_CALIBRATION_NUMBER_8</i>	calibrate 8 times

ADC_CALIBRATION_NUM16	calibrate 16 times
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC0 calibration number */
adc_calibration_number(ADC0, ADC_CALIBRATION_NUM1);
```

### adc\_calibration\_enable

The description of adc\_calibration\_enable is shown as below:

**Table 3-12. 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
ADCx(x=0,1,2,3)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* ADC0 calibration and reset calibration */
adc_calibration_enable(ADC0);
```

### adc\_resolution\_config

The description of adc\_resolution\_config is shown as below:

**Table 3-13. Function adc\_resolution\_config**

Function name	adc_resolution_config
Function prototype	void adc_resolution_config(uint32_t adc_periph, uint32_t resolution);
Function descriptions	configure ADC resolution
Precondition	-
The called functions	-



Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
Input parameter{in}	
<b>resolution</b>	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\_internal\_channel\_config

The description of adc\_internal\_channel\_config is shown as below:

**Table 3-14. Function adc\_internal\_channel\_config**

<b>Function name</b>	adc_internal_channel_config
<b>Function prototype</b>	void adc_internal_channel_config(uint32_t internal_channel, ControlStatus newvalue);
<b>Function descriptions</b>	enable or disable ADC internal channels
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>internal_channel</b>	the internal channels
<i>ADC_CHANNEL_INTE RNAL_TEMPSENSOR</i>	temperature sensor channel
<i>ADC_CHANNEL_INTE RNAL_VREFINT</i>	vrefint channel
<i>ADC_CHANNEL_INTE RNAL_VBAT</i>	vbat channel
<i>ADC_CHANNEL_INTE RNAL_HP_TEMPSENS</i>	high-precision temperature sensor channel

OR	
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC temperature sensor channel */
```

```
adc_internal_channel_config (ADC_CHANNEL_INTERNAL_TEMPSENSOR, ENABLE);
```

### adc\_gain\_mode\_enable

The description of adc\_gain\_mode\_enable is shown as below:

**Table 3-15. Function adc\_gain\_mode\_enable**

Function name	adc_gain_mode_enable
Function prototype	void adc_gain_mode_enable (uint32_t adc_periph);
Function descriptions	enable ADC gain mode
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC0 gain mode */
```

```
adc_gain_mode_enable (ADC0);
```

### adc\_gain\_mode\_disable

The description of adc\_gain\_mode\_disable is shown as below:

**Table 3-16. Function adc\_gain\_mode\_disable**

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

<b>Function descriptions</b>	disable ADC gain mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC0 gain mode */
adc_gain_mode_disable (ADC0);
```

### adc\_gain\_factor\_set

The description of adc\_gain\_factor\_set is shown as below:

**Table 3-17. Function adc\_gain\_factor\_set**

<b>Function name</b>	adc_gain_factor_set
<b>Function prototype</b>	void adc_gain_factor_set (uint32_t adc_periph, uint32_t factor);
<b>Function descriptions</b>	configure ADC gain factor
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>factor</b>	gain factor, 0..0x3FFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 gain factor */
adc_gain_factor_set (ADC0,0x1);
```

### adc\_dma\_mode\_enable

The description of adc\_dma\_mode\_enable is shown as below:

Table 3-18. Function `adc_dma_mode_enable`

<b>Function name</b>	<code>adc_dma_mode_enable</code>
<b>Function prototype</b>	<code>void adc_dma_mode_enable(uint32_t adc_periph);</code>
<b>Function descriptions</b>	enable ADC DMA request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-19. Function `adc_dma_mode_disable`

<b>Function name</b>	<code>adc_dma_mode_disable</code>
<b>Function prototype</b>	<code>void adc_dma_mode_disable(uint32_t adc_periph);</code>
<b>Function descriptions</b>	disable ADC DMA request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### **`adc_dma_request_after_last_enable`**

The description of `adc_dma_request_after_last_enable` is shown as below:

Table 3-20. Function `adc_dma_request_after_last_enable`

<b>Function name</b>	<code>adc_dma_request_after_last_enable</code>
<b>Function prototype</b>	<code>void adc_dma_request_after_last_enable(uint32_t adc_periph);</code>
<b>Function descriptions</b>	when DMA=1, the DMA engine issues a request at end of each routine conversion
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* when DMA=1, the DMA engine issues a request at end of each routine conversion for ADC0 */
```

```
adc_dma_request_after_last_enable(ADC0);
```

### **`adc_dma_request_after_last_disable`**

The description of `adc_dma_request_after_last_disable` is shown as below:

Table 3-21. Function `adc_dma_request_after_last_disable`

<b>Function name</b>	<code>adc_dma_request_after_last_disable</code>
<b>Function prototype</b>	<code>void adc_dma_request_after_last_disable(uint32_t adc_periph);</code>
<b>Function descriptions</b>	the DMA engine is disabled after the end of transfer signal from DMA controller is detected
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* the DMA engine is disabled after the end of transfer signal from DMA controller is detected for ADC0 */
```

```
adc_dma_request_after_last_disable (ADC0);
```

## adc\_hpdf\_mode\_enable

The description of adc\_hpdf\_mode\_enable is shown as below:

**Table 3-22. Function adc\_hpdf\_mode\_enable**

<b>Function name</b>	adc_hpdf_mode_enable
<b>Function prototype</b>	void adc_hpdf_mode_enable(uint32_t adc_periph);
<b>Function descriptions</b>	enable hpdf mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ADC0 hpdf mode */
adc_hpdf_mode_enable(ADC0);
```

## adc\_hpdf\_mode\_disable

The description of adc\_hpdf\_mode\_disable is shown as below:

**Table 3-23. Function adc\_hpdf\_mode\_disable**

<b>Function name</b>	adc_hpdf_mode_disable
<b>Function prototype</b>	void adc_hpdf_mode_disable(uint32_t adc_periph);
<b>Function descriptions</b>	disable hpdf mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC0 hpdf mode */
adc_hpdf_mode_disable(ADC0);
```

## adc\_discontinuous\_mode\_config

The description of adc\_discontinuous\_mode\_config is shown as below:

**Table 3-24. Function adc\_discontinuous\_mode\_config**

<b>Function name</b>	adc_discontinuous_mode_config
<b>Function prototype</b>	void adc_discontinuous_mode_config(uint32_t adc_periph, uint8_t adc_sequence, uint32_t length);
<b>Function descriptions</b>	configure ADC discontinuous mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_sequence</b>	select the channel sequence
ADC_ROUTINE_CHANNEL	routine channel sequence
ADC_INSERTED_CHANNEL	inserted channel sequence
ADC_CHANNEL_DISCON_DISABLE	disable discontinuous mode of routine and inserted sequence
<b>Input parameter{in}</b>	
<b>length</b>	number of conversions in discontinuous mode, the number can be 1..8 for routine channel, the number has no effect for inserted channel
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 routine channel sequence discontinuous mode */
adc_discontinuous_mode_config(ADC0, ADC_ROUTINE_CHANNEL, 6);
```

## adc\_channel\_length\_config

The description of adc\_channel\_length\_config is shown as below:

**Table 3-25. Function adc\_channel\_length\_config**

<b>Function name</b>	adc_channel_length_config
<b>Function prototype</b>	void adc_channel_length_config(uint32_t adc_periph, uint8_t adc_sequence, uint32_t length);
<b>Function descriptions</b>	configure the length of routine channel sequence or inserted channel sequence



<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_sequence</b>	select the channel sequence
<i>ADC_ROUTINE_CHANNEL</i>	routine channel sequence
<i>ADC_INSERTED_CHANNEL</i>	inserted channel sequence
<b>Input parameter{in}</b>	
<b>length</b>	the length of the channel, routine channel 1-16, inserted channel 1-4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### adc\_routine\_channel\_config

The description of adc\_routine\_channel\_config is shown as below:

**Table 3-26. Function adc\_routine\_channel\_config**

<b>Function name</b>	adc_routine_channel_config
<b>Function prototype</b>	void adc_routine_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
<b>Function descriptions</b>	configure ADC routine channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>rank</b>	the routine sequence rank, this parameter must be between 0 to 15
<b>Input parameter{in}</b>	
<b>adc_channel</b>	the selected ADC channel
<i>ADC_CHANNEL_x</i> (x=0..21)	ADC channelx
<b>Input parameter{in}</b>	

<b>sample_time</b>	the sample time value, 0..638
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 routine channel */
```

```
adc_routine_channel_config(ADC0, 1, ADC_CHANNEL_0, 10);
```

### adc\_inserted\_channel\_config

The description of adc\_inserted\_channel\_config is shown as below:

**Table 3-27. Function adc\_inserted\_channel\_config**

<b>Function name</b>	adc_inserted_channel_config
<b>Function prototype</b>	void adc_inserted_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
<b>Function descriptions</b>	configure ADC inserted channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>rank</b>	the inserted sequence rank, this parameter must be between 0 to 3
<b>Input parameter{in}</b>	
<b>adc_channel</b>	the selected ADC channel
ADC_CHANNEL_x (x=0..21)	ADC Channelx
<b>Input parameter{in}</b>	
<b>sample_time</b>	the sample time value, 0..638
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_inserted_channel_config(ADC0, 1, ADC_CHANNEL_0, 10);
```

## adc\_inserted\_channel\_offset\_config

The description of adc\_inserted\_channel\_offset\_config is shown as below:

**Table 3-28. Function adc\_inserted\_channel\_offset\_config**

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

Example:

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

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

## adc\_inserted\_channel\_mean\_value\_mode\_enable

The description of adc\_inserted\_channel\_mean\_value\_mode\_enable is shown as below:

**Table 3-29. Function adc\_inserted\_channel\_mean\_value\_mode\_enable**

<b>Function name</b>	adc_inserted_channel_mean_value_mode_enable
<b>Function prototype</b>	void adc_inserted_channel_mean_value_mode_enable (uint32_t adc_periph);
<b>Function descriptions</b>	enable mean value mode for inserted channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* enable ADC0 mean value mode for inserted channel */
adc_inserted_channel_mean_value_mode_enable (ADC0);
```

### adc\_inserted\_channel\_mean\_value\_mode\_disable

The description of adc\_inserted\_channel\_mean\_value\_mode\_disable is shown as below:

**Table 3-30. Function adc\_inserted\_channel\_mean\_value\_mode\_disable**

<b>Function name</b>	adc_inserted_channel_mean_value_mode_disable
<b>Function prototype</b>	void adc_inserted_channel_mean_value_mode_disable (uint32_t adc_periph);
<b>Function descriptions</b>	disable mean value mode for inserted channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<b>ADCx(x=0,1,2,3)</b>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC0 mean value mode for inserted channel */
adc_inserted_channel_mean_value_mode_disable (ADC0);
```

### adc\_inserted\_channel\_mean\_value\_mode\_config

The description of adc\_inserted\_channel\_mean\_value\_mode\_config is shown as below:

**Table 3-31. Function adc\_inserted\_channel\_mean\_value\_mode\_config**

<b>Function name</b>	adc_inserted_channel_mean_value_mode_config
<b>Function prototype</b>	void adc_inserted_channel_mean_value_mode_config (uint32_t adc_periph, uint32_t mode);
<b>Function descriptions</b>	configure mean value mode for inserted channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>mode</b>	mean value mode select
ADC_MEAN_VALUE_MODE_4X	4x mode
ADC_MEAN_VALUE_MODE_8X	8x mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 mean value mode for inserted channel */
```

```
adc_inserted_channel_mean_value_mode_config (ADC0, ADC_MEAN_VALUE_MODE_4X);
```

### adc\_channel\_differential\_mode\_config

The description of adc\_channel\_differential\_mode\_config is shown as below:

**Table 3-32. Function adc\_channel\_differential\_mode\_config**

<b>Function name</b>	adc_channel_differential_mode_config
<b>Function prototype</b>	void adc_channel_differential_mode_config(uint32_t adc_periph, uint32_t adc_channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure differential mode for ADC channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_channel</b>	the channel use differential mode
ADC_DIFFERENTIAL_MODE_CHANNEL_x (x=0..21), ADC_DIFFERENTIAL_MODE_CHANNEL_ALL	ADC channel for differential mode
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
ENABLE	enable function
DISABLE	disable function
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure differential mode for ADC channel */
```

```
adc_channel_differential_mode_config(ADC0,  
ADC_DIFFERENTIAL_MODE_CHANNEL_ALL, ENABLE);
```

### adc\_external\_trigger\_config

The description of adc\_external\_trigger\_config is shown as below:

**Table 3-33. Function adc\_external\_trigger\_config**

<b>Function name</b>	adc_external_trigger_config
<b>Function prototype</b>	void adc_external_trigger_config(uint32_t adc_periph, uint8_t adc_sequence, uint32_t trigger_mode)
<b>Function descriptions</b>	configure ADC external trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_sequence</b>	select the sequence
ADC_ROUTINE_CHANNEL	routine sequence
ADC_INSERTED_CHANNEL	inserted sequence
<b>Input parameter{in}</b>	
<b>trigger_mode</b>	external trigger mode
EXTERNAL_TRIGGER_DISABLE	external trigger disable
EXTERNAL_TRIGGER_RISING	rising edge of external trigger
EXTERNAL_TRIGGER_FALLING	falling edge of external trigger
EXTERNAL_TRIGGER_RISING_FALLING	rising and falling edge of external trigger
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config ADC0 inserted sequence external trigger */

adc_external_trigger_config(ADC0,ADC_INSERTED_CHANNEL,
EXTERNAL_TRIGGER_RISING);
```

### adc\_software\_trigger\_enable

The description of adc\_software\_trigger\_enable is shown as below:

**Table 3-34. Function adc\_software\_trigger\_enable**

<b>Function name</b>	adc_software_trigger_enable
<b>Function prototype</b>	void adc_software_trigger_enable(uint32_t adc_periph, uint8_t adc_sequence);
<b>Function descriptions</b>	enable ADC software trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_sequence</b>	select the sequence
ADC_ROUTINE_CHANNEL	routine sequence
ADC_INSERTED_CHANNEL	inserted sequence
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ADC0 routine sequence software trigger */

adc_software_trigger_enable(ADC0, ADC_ROUTINE_CHANNEL);
```

### adc\_end\_of\_conversion\_config

The description of adc\_end\_of\_conversion\_config is shown as below:

**Table 3-35. Function adc\_end\_of\_conversion\_config**

<b>Function name</b>	adc_end_of_conversion_config
<b>Function prototype</b>	void adc_end_of_conversion_config(uint32_t adc_periph, uint32_t end_selection);
<b>Function descriptions</b>	configure end of conversion mode

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>end_selection</b>	end of conversion mode
<i>ADC_EOC_SET_SEQUENCE</i>	only at the end of a sequence of routine conversions, the EOC bit is set. Overflow detection is disabled unless DMA=1
<i>ADC_EOC_SET_CONVERSION</i>	at the end of each routine conversion, the EOC bit is set. Overflow is detected automatically
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 end of conversion mode */
```

```
adc_end_of_conversion_config(ADC0, ADC_EOC_SET_SEQUENCE);
```

### adc\_routine\_data\_read

The description of adc\_routine\_data\_read is shown as below:

**Table 3-36. Function adc\_routine\_data\_read**

<b>Function name</b>	adc_routine_data_read
<b>Function prototype</b>	uint32_t adc_routine_data_read(uint32_t adc_periph);
<b>Function descriptions</b>	read ADC routine data register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	ADC conversion value

Example:

```
/* read ADC0 routine data register */
```

```
uint32_t adc_value = 0;
```

```
adc_value = adc_routine_data_read(ADC0);
```



## adc\_inserted\_data\_read

The description of adc\_inserted\_data\_read is shown as below:

**Table 3-37. Function adc\_inserted\_data\_read**

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

Example:

```
/* read ADC0 inserted data register */
uint32_t adc_value = 0;
adc_value = adc_inserted_data_read (ADC0, ADC_INSERTED_CHANNEL_0);
```

## adc\_watchdog0\_single\_channel\_enable

The description of adc\_watchdog0\_single\_channel\_enable is shown as below:

**Table 3-38. Function adc\_watchdog0\_single\_channel\_enable**

<b>Function name</b>	adc_watchdog0_single_channel_enable
<b>Function prototype</b>	void adc_watchdog0_single_channel_enable(uint32_t adc_periph, uint8_t adc_channel);
<b>Function descriptions</b>	configure ADC analog watchdog 0 single channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	

<b>adc_channel</b>	the selected ADC channel
<i>ADC_CHANNEL_x</i> (x=0..21)	ADC channelx
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### adc\_watchdog0\_sequence\_channel\_enable

The description of adc\_watchdog0\_sequence\_channel\_enable is shown as below:

**Table 3-39. Function adc\_watchdog0\_sequence\_channel\_enable**

<b>Function name</b>	adc_watchdog0_sequence_channel_enable
<b>Function prototype</b>	void adc_watchdog0_sequence_channel_enable(uint32_t adc_periph, uint8_t adc_sequence);
<b>Function descriptions</b>	configure ADC analog watchdog 0 sequence channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_sequence</b>	the sequence use analog watchdog 0
<i>ADC_ROUTINE_CHANNEL</i>	routine sequence
<i>ADC_INSERTED_CHANNEL</i>	inserted sequence
<i>ADC_ROUTINE_INSERTED_CHANNEL</i>	both routine and inserted sequence
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_watchdog0_sequence_channel_enable (ADC0, ADC_ROUTINE_CHANNEL);
```

## adc\_watchdog0\_disable

The description of adc\_watchdog0\_disable is shown as below:

**Table 3-40. Function adc\_watchdog0\_disable**

<b>Function name</b>	adc_watchdog0_disable
<b>Function prototype</b>	void adc_watchdog0_disable(uint32_t adc_periph);
<b>Function descriptions</b>	disable ADC analog watchdog 0
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-41. Function adc\_watchdog1\_channel\_config**

<b>Function name</b>	adc_watchdog1_channel_config
<b>Function prototype</b>	void adc_watchdog1_channel_config(uint32_t adc_periph, uint32_t selection_channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure ADC analog watchdog 1 channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>selection_channel</b>	the channel use analog watchdog 1
ADC_AWD1_2_SELEC TION_CHANNEL_x (x=0..21), ADC_AWD1_2_SELEC TION_CHANNEL_ALL	ADC channel analog watchdog 1 selection
<b>Input parameter{in}</b>	

<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### adc\_watchdog2\_channel\_config

The description of adc\_watchdog2\_channel\_config is shown as below:

**Table 3-42. Function adc\_watchdog2\_channel\_config**

<b>Function name</b>	adc_watchdog2_channel_config
<b>Function prototype</b>	void adc_watchdog2_channel_config(uint32_t adc_periph, uint32_t selection_channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure ADC analog watchdog 2 channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>selection_channel</b>	the channel use analog watchdog 2
<i>ADC_AWD1_2_SELECTION_CHANNEL_x</i> ( <i>x=0..21</i> ), <i>ADC_AWD1_2_SELECTION_CHANNEL_ALL</i>	ADC channel analog watchdog 2 selection
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC analog watchdog 2 channel */
```

```
adc_watchdog2_channel_config(ADC0, ADC_AWD1_SELECTION_CHANNEL_1,
ENABLE);
```

### adc\_watchdog1\_disable

The description of adc\_watchdog1\_disable is shown as below:

**Table 3-43. Function adc\_watchdog1\_disable**

<b>Function name</b>	adc_watchdog1_disable
<b>Function prototype</b>	void adc_watchdog1_disable(uint32_t adc_periph);
<b>Function descriptions</b>	disable ADC analog watchdog 1
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC0 analog watchdog 1 */
```

```
adc_watchdog1_disable(ADC0);
```

### adc\_watchdog2\_disable

The description of adc\_watchdog2\_disable is shown as below:

**Table 3-44. Function adc\_watchdog2\_disable**

<b>Function name</b>	adc_watchdog2_disable
<b>Function prototype</b>	void adc_watchdog2_disable(uint32_t adc_periph);
<b>Function descriptions</b>	disable ADC analog watchdog 2
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC0 analog watchdog 2 */
```

```
adc_watchdog2_disable(ADC0);
```

### adc\_watchdog0\_threshold\_config

The description of adc\_watchdog0\_threshold\_config is shown as below:

**Table 3-45. Function adc\_watchdog0\_threshold\_config**

<b>Function name</b>	adc_watchdog0_threshold_config
<b>Function prototype</b>	void adc_watchdog0_threshold_config(uint32_t adc_periph , uint32_t low_threshold , uint32_t high_threshold);
<b>Function descriptions</b>	configure ADC analog watchdog 0 threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>low_threshold</b>	analog watchdog 0 low threshold, 0..0x3FFFFFF
<b>Input parameter{in}</b>	
<b>high_threshold</b>	analog watchdog 0 high threshold, 0..0x3FFFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC2 analog watchdog 0 threshold */
```

```
adc_watchdog0_threshold_config(ADC2, 0x0400, 0xA00);
```

### adc\_watchdog1\_threshold\_config

The description of adc\_watchdog1\_threshold\_config is shown as below:

**Table 3-46. Function adc\_watchdog1\_threshold\_config**

<b>Function name</b>	adc_watchdog1_threshold_config
<b>Function prototype</b>	void adc_watchdog1_threshold_config(uint32_t adc_periph , uint32_t low_threshold , uint32_t high_threshold);
<b>Function descriptions</b>	configure ADC analog watchdog 1 threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>low_threshold</b>	analog watchdog 1 low threshold, 0..0x3FFFFFF
<b>Input parameter{in}</b>	
<b>high_threshold</b>	analog watchdog 1 high threshold, 0..0x3FFFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC2 analog watchdog 1 threshold */
adc_watchdog1_threshold_config(ADC2, 0x40, 0xA0);
```

### adc\_watchdog2\_threshold\_config

The description of adc\_watchdog2\_threshold\_config is shown as below:

**Table 3-47. Function adc\_watchdog2\_threshold\_config**

<b>Function name</b>	adc_watchdog2_threshold_config
<b>Function prototype</b>	void adc_watchdog2_threshold_config(uint32_t adc_periph , uint32_t low_threshold , uint32_t high_threshold);
<b>Function descriptions</b>	configure ADC analog watchdog 2 threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>low_threshold</b>	analog watchdog 2 low threshold, 0..0x3FFFFFF
<b>Input parameter{in}</b>	
<b>high_threshold</b>	analog watchdog 2 high threshold, 0..0x3FFFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC2 analog watchdog 2 threshold */
adc_watchdog2_threshold_config(ADC2, 0x40, 0xA0);
```

## adc\_oversample\_mode\_config

The description of adc\_oversample\_mode\_config is shown as below:

**Table 3-48. Function adc\_oversample\_mode\_config**

<b>Function name</b>	adc_oversample_mode_config
<b>Function prototype</b>	void adc_oversample_mode_config(uint32_t adc_periph, uint32_t mode, uint16_t shift, uint16_t ratio);
<b>Function descriptions</b>	configure ADC oversample mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>mode</b>	ADC oversampling mode
ADC_OVERSAMPLING _ALL_CONVERT	all oversampled conversions for a channel are done consecutively after a trigger
ADC_OVERSAMPLING _ONE_CONVERT	each oversampled conversion for a channel needs a trigger
<b>Input parameter{in}</b>	
<b>shift</b>	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
ADC_OVERSAMPLING _SHIFT_9B	9-bit oversampling shift
ADC_OVERSAMPLING	10-bit oversampling shift



<code>_SHIFT_10B</code>	
<code>ADC_OVERSAMPLING</code> <code>_SHIFT_11B</code>	11-bit oversampling shift
<b>Input parameter{in}</b>	
<b>ratio</b>	ADC oversampling ratio, 0..1023
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_oversample_mode_config(ADC1, ADC_OVERSAMPLING_ALL_CONVERT,
ADC_OVERSAMPLING_SHIFT_4B, 15);
```

### adc\_oversample\_mode\_enable

The description of `adc_oversample_mode_enable` is shown as below:

**Table 3-49. Function `adc_oversample_mode_enable`**

<b>Function name</b>	<code>adc_oversample_mode_enable</code>
<b>Function prototype</b>	<code>void adc_oversample_mode_enable(uint32_t adc_periph);</code>
<b>Function descriptions</b>	enable ADC oversample mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<code>ADCx(x=0,1,2,3)</code>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-50. Function `adc_oversample_mode_disable`**

<b>Function name</b>	<code>adc_oversample_mode_disable</code>
----------------------	--

<b>Function prototype</b>	void adc_oversample_mode_disable(uint32_t adc_periph);
<b>Function descriptions</b>	disable ADC oversample mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_oversample_mode_disable (ADC0);
```

### adc\_flag\_get

The description of adc\_flag\_get is shown as below:

**Table 3-51. Function adc\_flag\_get**

<b>Function name</b>	adc_flag_get
<b>Function prototype</b>	FlagStatus adc_flag_get(uint32_t adc_periph, uint32_t flag);
<b>Function descriptions</b>	get the ADC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>flag</b>	the adc flag bits
<i>ADC_FLAG_WDE0</i>	analog watchdog 0 event flag
<i>ADC_FLAG_EOC</i>	end of sequence conversion flag
<i>ADC_FLAG_EOIC</i>	end of inserted sequence conversion flag
<i>ADC_FLAG_STIC</i>	start flag of inserted sequence
<i>ADC_FLAG_STRC</i>	start flag of routine sequence
<i>ADC_FLAG_ROVF</i>	routine data register overflow flag
<i>ADC_FLAG_WDE1</i>	analog watchdog 1 event flag
<i>ADC_FLAG_WDE2</i>	analog watchdog 2 event flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-52. Function adc\_flag\_clear**

<b>Function name</b>	adc_flag_clear
<b>Function prototype</b>	void adc_flag_clear(uint32_t adc_periph, uint32_t flag);
<b>Function descriptions</b>	clear the ADC flag bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>flag</b>	the adc flag bits
ADC_FLAG_WDE0	analog watchdog 0 event flag
ADC_FLAG_EOC	end of sequence conversion flag
ADC_FLAG_EOIC	end of inserted sequence conversion flag
ADC_FLAG_STIC	start flag of inserted sequence
ADC_FLAG_STRC	start flag of routine sequence
ADC_FLAG_ROVF	routine data register overflow flag
ADC_FLAG_WDE1	analog watchdog 1 event flag
ADC_FLAG_WDE2	analog watchdog 2 event flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-53. Function `adc_interrupt_enable`

<b>Function name</b>	<code>adc_interrupt_enable</code>
<b>Function prototype</b>	<code>void adc_interrupt_enable(uint32_t adc_periph, uint32_t interrupt);</code>
<b>Function descriptions</b>	enable ADC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_interrupt</b>	the adc interrupt
<i>ADC_INT_WDE0</i>	analog watchdog 0 interrupt
<i>ADC_INT_EOC</i>	end of sequence conversion interrupt
<i>ADC_INT_EOIC</i>	end of inserted sequence conversion interrupt
<i>ADC_INT_ROVF</i>	routine data register overflow interrupt
<i>ADC_INT_WDE1</i>	analog watchdog 1 interrupt
<i>ADC_INT_WDE2</i>	analog watchdog 2 interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-54. Function `adc_interrupt_disable`

<b>Function name</b>	<code>adc_interrupt_disable</code>
<b>Function prototype</b>	<code>void adc_interrupt_disable(uint32_t adc_periph , uint32_t interrupt);</code>
<b>Function descriptions</b>	disable ADC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_interrupt</b>	the adc interrupt
<i>ADC_INT_WDE0</i>	analog watchdog 0 interrupt
<i>ADC_INT_EOC</i>	end of sequence conversion interrupt

<i>ADC_INT_EOIC</i>	end of inserted sequence conversion interrupt
<i>ADC_INT_ROVF</i>	routine data register overflow interrupt
<i>ADC_INT_WDE1</i>	analog watchdog 1 interrupt
<i>ADC_INT_WDE2</i>	analog watchdog 2 interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-55. Function `adc_interrupt_flag_get`**

<b>Function name</b>	<code>adc_interrupt_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus adc_interrupt_flag_get(uint32_t adc_periph, uint32_t int_flag);</code>
<b>Function descriptions</b>	get the ADC interrupt bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2,3)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>int_flag</b>	the adc interrupt bits
<i>ADC_INT_FLAG_WDE0</i>	analog watchdog 0 interrupt
<i>ADC_INT_FLAG_EOC</i>	end of sequence conversion interrupt
<i>ADC_INT_FLAG_EOIC</i>	end of inserted sequence conversion interrupt
<i>ADC_INT_FLAG_ROV</i>	routine data register overflow interrupt
<i>ADC_INT_FLAG_WDE1</i>	analog watchdog 1 interrupt
<i>ADC_INT_FLAG_WDE2</i>	analog watchdog 2 interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

```
FlagStatus flag_value;
```

```
flag_value = adc_interrupt_flag_get(ADC0, ADC_INT_FLAG_WDE0);
```

### adc\_interrupt\_flag\_clear

The description of adc\_interrupt\_flag\_clear is shown as below:

**Table 3-56. Function adc\_interrupt\_flag\_clear**

<b>Function name</b>	adc_interrupt_flag_clear
<b>Function prototype</b>	void adc_interrupt_flag_clear(uint32_t adc_periph, uint32_t int_flag);
<b>Function descriptions</b>	clear the ADC interrupt bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2,3)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>int_flag</b>	the adc interrupt bits
ADC_INT_FLAG_WDE0	analog watchdog 0 interrupt flag
ADC_INT_FLAG_EOC	end of sequence conversion interrupt flag
ADC_INT_FLAG_EOIC	end of inserted sequence conversion interrupt flag
ADC_INT_FLAG_ROV F	routine data register overflow interrupt flag
ADC_INT_FLAG_WDE1	analog watchdog 1 interrupt flag
ADC_INT_FLAG_WDE2	analog watchdog 2 interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_interrupt_flag_clear(ADC0, ADC_INT_WDE0);
```

### adc\_sync\_mode\_config

The description of adc\_sync\_mode\_config is shown as below:

Table 3-57. Function `adc_sync_mode_config`

<b>Function name</b>	<code>adc_sync_mode_config</code>
<b>Function prototype</b>	<code>void adc_sync_mode_config(uint32_t sync_mode);</code>
<b>Function descriptions</b>	configure the ADC sync mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sync_mode</b>	ADC sync mode
<code>ADC_SYNC_MODE_INDEPENDENT</code>	all the ADCs work independently
<code>ADC_DUAL_REGULAR_PARALLEL_INSERTED_PARALLEL</code>	ADC0 and ADC1 work in combined routine parallel & inserted parallel mode
<code>ADC_DUAL_REGULAR_PARALLEL_INSERTED_ROTATION</code>	ADC0 and ADC1 work in combined routine parallel & trigger rotation mode
<code>ADC_DUAL_INSERTED_PARALLEL</code>	ADC0 and ADC1 work in inserted parallel mode
<code>ADC_DUAL_ROUTINE_PARALLEL</code>	ADC0 and ADC1 work in routine parallel mode
<code>ADC_DUAL_ROUTINE_FOLLOW_UP</code>	ADC0 and ADC1 work in follow-up mode
<code>ADC_DUAL_INSERTED_TRIGGER_ROTATION</code>	ADC0 and ADC1 work in trigger rotation mode
<code>ADC_TRIPLE_ROUTINE_PARALLEL_INSERTED_PARALLEL</code>	ADC0/1/2 work in combined routine parallel & inserted parallel mode
<code>ADC_TRIPLE_ROUTINE_PARALLEL_INSERTED_ROTATION</code>	ADC0/1/2 work in combined routine parallel & trigger rotation mode
<code>ADC_TRIPLE_INSERTED_PARALLEL</code>	ADC0/1/2 work in inserted parallel mode
<code>ADC_TRIPLE_ROUTINE_PARALLEL</code>	ADC0/1/2 work in routine parallel mode

<i>E_PARALLEL</i>	
<i>ADC_TRIPLE_ROUTINE_FOLLOW_UP</i>	ADC0/1/2 work in follow-up mode
<i>ADC_TRIPLE_INSERTED_TRIGGER_ROTATION</i>	ADC0/1/2 work in trigger rotation mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* ADC0 and ADC1 work in combined routine parallel & inserted parallel mode */
```

```
adc_sync_mode_config(ADC_DUAL_REGULAR_PARALLEL_INSERTED_PARALLEL);
```

### adc\_sync\_delay\_config

The description of adc\_sync\_delay\_config is shown as below:

**Table 3-58. Function adc\_interrupt\_disable**

<b>Function name</b>	adc_sync_delay_config
<b>Function prototype</b>	void adc_sync_delay_config(uint32_t sample_delay);
<b>Function descriptions</b>	configure the delay between 2 sampling phases in ADC sync modes
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sample_delay</b>	the delay between 2 sampling phases in ADC sync modes
<i>ADC_SYNC_DELAY_x CYCLE(x=5..20)</i>	the delay between 2 sampling phases in ADC sync modes is x ADC clock cycles
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:



```
/* configure the delay between 2 sampling phases in ADC sync modes */
```

```
adc_sync_delay_config (ADC_SYNC_DELAY_5CYCLE);
```

### adc\_sync\_dma\_config

The description of adc\_sync\_dma\_config is shown as below:

**Table 3-59. Function adc\_sync\_dma\_config**

<b>Function name</b>	adc_sync_dma_config
<b>Function prototype</b>	void adc_sync_dma_config(uint32_t dma_mode );
<b>Function descriptions</b>	configure ADC sync DMA mode selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_mode</b>	ADC sync DMA mode
ADC_SYNC_DMA_DISABLE	ADC sync DMA disabled
ADC_SYNC_DMA_MODE0	ADC sync DMA mode 0
ADC_SYNC_DMA_MODE1	ADC sync DMA mode 1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC sync DMA mode selection */
```

```
adc_sync_dma_config (ADC_SYNC_DMA_MODE0);
```

### adc\_sync\_dma\_request\_after\_last\_enable

The description of adc\_sync\_dma\_request\_after\_last\_enable is shown as below:

**Table 3-60. Function adc\_sync\_dma\_request\_after\_last\_enable**

<b>Function name</b>	adc_sync_dma_request_after_last_enable
<b>Function prototype</b>	void adc_sync_dma_request_after_last_enable(void);
<b>Function descriptions</b>	when SYNC DMA is not equal to 2'b00, the DMA engine issues requests according to the SYNC DMA bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* when SYNC DMA is not equal to 2'b00, the DMA engine issues requests according to the SYNC DMA bits */
```

```
adc_sync_dma_request_after_last_enable();
```

### adc\_sync\_dma\_request\_after\_last\_disable

The description of `adc_sync_dma_request_after_last_disable` is shown as below:

**Table 3-61. Function `adc_sync_dma_request_after_last_disable`**

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

Example:

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

```
adc_sync_dma_request_after_last_disable();
```

### adc\_sync\_master\_adc\_routine\_data\_read

The description of `adc_sync_master_adc_routine_data_read` is shown as below:

**Table 3-62. Function `adc_sync_master_adc_routine_data_read`**

Function name	adc_sync_master_adc_routine_data_read
Function prototype	uint16_t adc_sync_master_adc_routine_data_read (void);
Function descriptions	read ADC sync master adc routine data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
uint16_t	sync routine data

Example:

```
/* read ADC sync master adc routine data register */
adc_sync_master_adc_routine_data_read ();
```

### adc\_sync\_slave\_adc\_routine\_data\_read

The description of adc\_sync\_slave\_adc\_routine\_data\_read is shown as below:

**Table 3-63. Function adc\_sync\_slave\_adc\_routine\_data\_read**

Function name	adc_sync_slave_adc_routine_data_read
Function prototype	uint16_t adc_sync_slave_adc_routine_data_read (void);
Function descriptions	read ADC sync slave adc routine data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	sync routine data

Example:

```
/* read ADC sync slave adc routine data register */
adc_sync_slave_adc_routine_data_read ();
```

## 3.3. 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 [3.3.1](#), the CAN firmware functions are introduced in chapter [3.3.2](#).

### 3.3.1. Descriptions of Peripheral registers

CAN registers are listed in the table shown as below:

Table 3-64. 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
CAN_FDCTL	CAN FD control register
CAN_FDBT	CAN bit timing register
CAN_CRCCFD	CAN CRC for classical and FD frame register

### 3.3.2. Descriptions of Peripheral functions

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

**Table 3-65. 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
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

Function name	Function description
	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_bsp_syn_config	configure bit sampling synchronization
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 3-66. 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 3-67. 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 3-68. 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[64]	data
data_bytes	data bytes
padding	FD mode padding data

### Structure can\_rx\_fifo\_struct

**Table 3-69. 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 3-70. Structure can\_fd\_parameter\_struct**

Member name	Function description
iso_can_fd_enable	ISO CAN FD protocol enable
irate_switch_enable	data bit rate switch
mailbox_data_size	mailbox data size
tdc_enable	trnasmitter delay compensation enable
tdc_offset	trnasmitter 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 3-71. 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 3-72. 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 3-73. 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 3-74. 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 3-75. 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 3-76. 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	bus off recovery interrupt

Member name	Function description
_RECOVERY	
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
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

Member name	Function description
CAN_INT_WAKEUP_MATCH	Pretended Networking match interrupt
CAN_INT_WAKEUP_TIMEOUT	Pretended Networking timeout wakeup interrupt

### Enum can\_flag\_enum

Table 3-77. 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
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

Member name	Function description
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
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

Member name	Function description
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 3-78. 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_RECOVERY	bus off recovery interrupt flag
CAN_INT_FLAG_ERROR_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

Member name	Function description
CAN_INT_FLAG_M B3	mailbox 3 interrupt flag
CAN_INT_FLAG_M B4	mailbox 4 interrupt flag
CAN_INT_FLAG_M B5	mailbox 5 interrupt flag
CAN_INT_FLAG_M B6	mailbox 6 interrupt flag
CAN_INT_FLAG_M B7	mailbox 7 interrupt flag
CAN_INT_FLAG_M B8	mailbox 8 interrupt flag
CAN_INT_FLAG_M B9	mailbox 9 interrupt flag
CAN_INT_FLAG_M B10	mailbox 10 interrupt flag
CAN_INT_FLAG_M B11	mailbox 11 interrupt flag
CAN_INT_FLAG_M B12	mailbox 12 interrupt flag
CAN_INT_FLAG_M B13	mailbox 13 interrupt flag
CAN_INT_FLAG_M B14	mailbox 14 interrupt flag
CAN_INT_FLAG_M B15	mailbox 15 interrupt flag
CAN_INT_FLAG_M B16	mailbox 16 interrupt flag
CAN_INT_FLAG_M B17	mailbox 17 interrupt flag
CAN_INT_FLAG_M B18	mailbox 18 interrupt flag
CAN_INT_FLAG_M B19	mailbox 19 interrupt flag
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

Member name	Function description
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 3-79. Enum can\_operation\_modes\_enum**

Member name	Function description
CAN_NORMAL_MO DE	normal mode
CAN_MONITOR_M ODE	monitor mode
CAN_LOOPBACK_ SILENT_MODE	loopback mode
CAN_INACTIVE_M ODE	inactive mode
CAN_DISABLE_MO DE	disable mode
CAN_PN_MODE	Pretended Networking mode

## Enum can\_struct\_type\_enum

**Table 3-80. Enum can\_struct\_type\_enum**

Member name	Function description
CAN_INIT_STRUCT	CAN initilaze 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 strcut
CAN_PN_MODE_FILTER_STRUCT	Pretended Networking mode filter parameter strcut
CAN_MDSC_STRUCT	mailbox descriptor strcut
CAN_FDES_STRUCT	Rx fifo descriptor strcut
CAN_FIFO_ID_FILTER_STRUCT	Rx fifo id filter strcut
CAN_CRC_STRUCT	CRC strcut
CAN_ERRCNT_STRUCT	error counter strcut

## Enum can\_error\_state\_enum

**Table 3-81. 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 3-82. Function can\_deinit**

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



<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### can\_software\_reset

The description of can\_software\_reset is shown as below:

**Table 3-83. Function can\_software\_reset**

<b>Function name</b>	can_software_reset
<b>Function prototype</b>	ErrStatus can_software_reset(uint32_t can_periph);
<b>Function descriptions</b>	reset CAN internal state machines and CAN registers
<b>Precondition</b>	-
<b>The called functions</b>	
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	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 3-84. Function can\_init**

<b>Function name</b>	can_init
<b>Function prototype</b>	ErrStatus can_init(uint32_t can_periph, can_parameter_struct*

	can_parameter_init);
<b>Function descriptions</b>	CAN module initialization
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0, 1, 2)</b>	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>can_parameter_init</b>	Refers to <a href="#">Table 3-67. Structure can_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	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 3-85. Function can\_struct\_para\_init**

<b>Function name</b>	can_struct_para_init
<b>Function prototype</b>	void can_struct_para_init(can_struct_type_enum type, void* p_struct);
<b>Function descriptions</b>	initialize CAN parameter structure with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>type</b>	Refers to enum <a href="#">Table 3-80. Enum can_struct_type_enum</a>
<b>Input parameter{in}</b>	
<b>p_struct</b>	the pointer of the specific struct
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-86. Function can\_private\_filter\_config**

<b>Function name</b>	can_private_filter_config
<b>Function prototype</b>	void can_private_filter_config(uint32_t can_periph, uint32_t index, uint32_t filter_data);
<b>Function descriptions</b>	configure receive fifo/mailbox private filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>index</b>	mailbox index
0..31	CAN mailbox index selection
<b>Input parameter{in}</b>	
<b>filter_data</b>	filter data to configure
0..0xFFFFFFFF	filter data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-87. Function can\_operation\_mode\_enter**

<b>Function name</b>	can_operation_mode_enter
<b>Function prototype</b>	ErrStatus can_operation_mode_enter(uint32_t can_periph, can_operation_modes_enum mode);
<b>Function descriptions</b>	enter the corresponding mode
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Input parameter{in}	
<b>mode</b>	Refers to enum <a href="#">Table 3-79. Enum can operation modes enum</a>
Output parameter{out}	
-	-
Return value	
<b>ErrStatus</b>	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 3-88. Function can\_operation\_mode\_get**

<b>Function name</b>	can_operation_mode_get
<b>Function prototype</b>	can_operation_modes_enum can_operation_mode_get(uint32_t can_periph);
<b>Function descriptions</b>	get operation mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
<b>can_operation_modes_enum</b>	Refers to enum <a href="#">Table 3-79. Enum can operation modes enum</a>

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 3-89. Function can\_inactive\_mode\_exit**

<b>Function name</b>	can_inactive_mode_exit
<b>Function prototype</b>	ErrStatus can_inactive_mode_exit(uint32_t can_periph);
<b>Function descriptions</b>	exit inactive mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	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 3-90. Function can\_pn\_mode\_exit**

<b>Function name</b>	can_pn_mode_exit
<b>Function prototype</b>	ErrStatus can_pn_mode_exit(uint32_t can_periph);
<b>Function descriptions</b>	exit Pretended Networking mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	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 3-91. Function can\_fd\_config**

<b>Function name</b>	can_fd_config
<b>Function prototype</b>	void can_fd_config(uint32_t can_periph, can_fd_parameter_struct *can_fd_para_init);
<b>Function descriptions</b>	can FD initialize
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>can_fd_para_init</b>	Refers to structure <a href="#">Table 3-70. Structure can fd parameter struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-92. Function can\_bitrate\_switch\_enable**

<b>Function name</b>	can_bitrate_switch_enable
<b>Function prototype</b>	void can_bitrate_switch_enable(uint32_t can_periph);
<b>Function descriptions</b>	enable bit rate switching
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	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 3-93. 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=0,1,2)	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 3-94. 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</i> ( <i>x</i> =0, 1, 2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	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 3-95. Function `can_tdc_enable`**

<b>Function name</b>	<code>can_tdc_enable</code>
<b>Function prototype</b>	<code>void can_tdc_enable(uint32_t can_periph);</code>
<b>Function descriptions</b>	enable transmitter delay compensation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx</i> ( <i>x</i> =0, 1, 2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
can_tdc_enable(CAN0);
```

### can\_tdc\_disable

The description of `can_tdc_disable` is shown as below:

**Table 3-96. Function `can_tdc_disable`**

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



Input parameter{in}	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</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 3-97. Function can\_rx\_fifo\_config**

<b>Function name</b>	can_rx_fifo_config
<b>Function prototype</b>	void can_rx_fifo_config(uint32_t can_periph, can_fifo_parameter_struct *can_fifo_para_init);
<b>Function descriptions</b>	configure rx FIFO
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Input parameter{in}	
<b>can_fifo_para_init</b>	Refers to structure <a href="#">Table 3-72. Structure can_fifo_parameter_struct</a>
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 3-98. 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=0,1,2)</code>	CAN peripheral selection
Input parameter{in}	
<code>id_filter_table</code>	Refers to structure <a href="#">Table 3-71. Structure <code>can_rx_fifo_id_filter_struct</code></a>
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 3-99. 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=0,1,2)</code>	CAN peripheral selection
Output parameter{out}	
<code>rx_fifo</code>	Refers to structure <a href="#">Table 3-69. Structure <code>can_rx_fifo_struct</code></a>
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 3-100. Function can\_rx\_fifo\_filter\_matching\_number\_get**

<b>Function name</b>	can_rx_fifo_filter_matching_number_get
<b>Function prototype</b>	uint32_t can_rx_fifo_filter_matching_number_get(uint32_t can_periph);
<b>Function descriptions</b>	get rx FIFO filter matching number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	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 3-101. Function can\_rx\_fifo\_clear**

<b>Function name</b>	can_rx_fifo_clear
<b>Function prototype</b>	void can_rx_fifo_clear(uint32_t can_periph);
<b>Function descriptions</b>	clear rx FIFO
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-

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 3-102. Function can\_ram\_address\_get**

<b>Function name</b>	can_ram_address_get
<b>Function prototype</b>	uint32_t can_ram_address_get(uint32_t can_periph, uint32_t index);
<b>Function descriptions</b>	get mailbox RAM address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>index</b>	mailbox index
0-31	mailbox index selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
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 3-103. Function can\_mailbox\_config**

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

The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure <a href="#">Table 3-68. Structure can_mailbox_descriptor_struct</a>
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 3-104. 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=0,1,2)	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 3-105. Function can\_mailbox\_transmit\_inactive**

<b>Function name</b>	can_mailbox_transmit_inactive
<b>Function prototype</b>	void can_mailbox_transmit_inactive(uint32_t can_periph, uint32_t index);
<b>Function descriptions</b>	inactive transmit mailbox
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>index</b>	mailbox index
0-31	mailbox index selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-106. Function can\_mailbox\_receive\_data\_read**

<b>Function name</b>	can_mailbox_receive_data_read
<b>Function prototype</b>	ErrStatus can_mailbox_receive_data_read(uint32_t can_periph, uint32_t index, can_mailbox_descriptor_struct *mdpara);
<b>Function descriptions</b>	read receive mailbox data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	

<b>index</b>	mailbox index
0-31	mailbox index selection
<b>Input parameter{in}</b>	
<b>mdpara</b>	Refers to structure <a href="#">Table 3-68. Structure can_mailbox_descriptor_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	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 3-107. Function can\_mailbox\_receive\_lock**

<b>Function name</b>	can_mailbox_receive_lock
<b>Function prototype</b>	void can_mailbox_receive_lock(uint32_t can_periph, uint32_t index);
<b>Function descriptions</b>	lock the receive mailbox
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>index</b>	mailbox index
0-31	mailbox index selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-108. Function can\_mailbox\_receive\_unlock**

<b>Function name</b>	can_mailbox_receive_unlock
<b>Function prototype</b>	void can_mailbox_receive_unlock(uint32_t can_periph);
<b>Function descriptions</b>	unlock the receive mailbox
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-109. Function can\_mailbox\_receive\_inactive**

<b>Function name</b>	can_mailbox_receive_inactive
<b>Function prototype</b>	void can_mailbox_receive_inactive(uint32_t can_periph, uint32_t index);
<b>Function descriptions</b>	inactive the receive mailbox
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>index</b>	mailbox index
0-31	mailbox index selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-110. Function can\_mailbox\_code\_get**

<b>Function name</b>	can_mailbox_code_get
<b>Function prototype</b>	uint32_t can_mailbox_code_get(uint32_t can_periph, uint32_t index);
<b>Function descriptions</b>	get mailbox code value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>index</b>	mailbox index
0-31	mailbox index selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
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 3-111. Function can\_error\_counter\_config**

<b>Function name</b>	can_error_counter_config
<b>Function prototype</b>	void can_error_counter_config(uint32_t can_periph, can_error_counter_struct *errcnt_struct);
<b>Function descriptions</b>	configure error counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection

Input parameter{in}	
<b>errcnt_struct</b>	Refers to structure <a href="#">Table 3-66. Structure can error counter struct</a>
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 3-112. Function can\_error\_counter\_get**

<b>Function name</b>	can_error_counter_get
<b>Function prototype</b>	void can_error_counter_get(uint32_t can_periph, can_error_counter_struct *errcnt_struct);
<b>Function descriptions</b>	get error count
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
<b>errcnt_struct</b>	Refers to structure <a href="#">Table 3-66. Structure can error counter struct</a>
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 3-113. 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=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
can_error_state_enum	Refers to enum <a href="#">Table 3-81. Enum can_error_state_enum</a>

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 3-114. 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=0,1,2)	CAN peripheral selection
Return value	
can_crc_struct	Refers to structure <a href="#">Table 3-75. Structure can_crc_struct</a>
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 3-115. Function can\_pn\_mode\_config**

<b>Function name</b>	can_pn_mode_config
<b>Function prototype</b>	void can_pn_mode_config(uint32_t can_periph, can_pn_mode_config_struct *pnmod_config);
<b>Function descriptions</b>	configure Pretended Networking mode parameter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Return value</b>	
<b>pnmod_config</b>	Refers to structure <a href="#">Table 3-74. Structure can_pn_mode_config_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-116. Function can\_pn\_mode\_filter\_config**

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

<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Return value	
<b>expect</b>	Refers to structure <a href="#">Table 3-73. Structure can_pn_mode_filter_struct</a>
Return value	
<b>filter</b>	Refers to structure <a href="#">Table 3-73. Structure can_pn_mode_filter_struct</a>
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 3-117. Function can\_pn\_mode\_num\_of\_match\_get**

<b>Function name</b>	can_pn_mode_num_of_match_get
<b>Function prototype</b>	int32_t can_pn_mode_num_of_match_get(uint32_t can_periph);
<b>Function descriptions</b>	get matching message counter of Pretended Networking mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
<b>int32_t</b>	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 3-118. 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=0,1,2)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure <a href="#">Table 3-68. Structure can_mailbox_descriptor_struct</a>
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 3-119. 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=0,1,2)	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 3-120. Function can\_self\_reception\_disable**

<b>Function name</b>	can_self_reception_disable
<b>Function prototype</b>	void can_self_reception_disable(uint32_t can_periph);
<b>Function descriptions</b>	disable self reception
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-121. Function can\_transmit\_abort\_enable**

<b>Function name</b>	can_transmit_abort_enable
<b>Function prototype</b>	void can_transmit_abort_enable(uint32_t can_periph);
<b>Function descriptions</b>	enable transmit abort
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-122. Function can\_transmit\_abort\_disable**

<b>Function name</b>	can_transmit_abort_disable
<b>Function prototype</b>	void can_transmit_abort_disable(uint32_t can_periph);
<b>Function descriptions</b>	disable transmit abort
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-123. Function can\_auto\_busoff\_recovery\_enable**

<b>Function name</b>	can_auto_busoff_recovery_enable
<b>Function prototype</b>	void can_auto_busoff_recovery_enable(uint32_t can_periph);
<b>Function descriptions</b>	enable auto bus off recovery mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-



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 3-124. Function can\_auto\_busoff\_recovery\_disable**

<b>Function name</b>	can_auto_busoff_recovery_disable
<b>Function prototype</b>	void can_auto_busoff_recovery_disable(uint32_t can_periph);
<b>Function descriptions</b>	disable auto bus off recovery mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-125. Function can\_time\_sync\_enable**

<b>Function name</b>	can_time_sync_enable
<b>Function prototype</b>	void can_time_sync_enable(uint32_t can_periph);
<b>Function descriptions</b>	enable time sync mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-126. Function can\_time\_sync\_disable**

<b>Function name</b>	can_time_sync_disable
<b>Function prototype</b>	void can_time_sync_disable(uint32_t can_periph);
<b>Function descriptions</b>	disable time sync mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-127. Function can\_edge\_filter\_mode\_enable**

<b>Function name</b>	can_edge_filter_mode_enable
<b>Function prototype</b>	void can_edge_filter_mode_enable(uint32_t can_periph);
<b>Function descriptions</b>	enable edge filter mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-128. Function can\_edge\_filter\_mode\_disable**

<b>Function name</b>	can_edge_filter_mode_disable
<b>Function prototype</b>	void can_edge_filter_mode_disable(uint32_t can_periph);
<b>Function descriptions</b>	disable edge filter mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-129. Function can\_ped\_mode\_enable**

<b>Function name</b>	can_ped_mode_enable
<b>Function prototype</b>	void can_ped_mode_enable(uint32_t can_periph);
<b>Function descriptions</b>	enable protocol exception detection mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-130. Function can\_ped\_mode\_disable**

<b>Function name</b>	can_ped_mode_disable
<b>Function prototype</b>	void can_ped_mode_disable(uint32_t can_periph);
<b>Function descriptions</b>	disable protocol exception detection mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-131. Function can\_arbitration\_delay\_bits\_config**

<b>Function name</b>	can_arbitration_delay_bits_config
<b>Function prototype</b>	void can_arbitration_delay_bits_config(uint32_t can_periph, uint32_t delay_bits);
<b>Function descriptions</b>	configure arbitration delay bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>delay_bits</b>	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 3-132. 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
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
sampling_mode	bsp sample mode
CAN_BSP_MODE_ON E_SAMPLE	one sample for received bit
CAN_BSP_MODE_TR HEE_SAMPLES	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\_bsp\_syn\_config

The description of can\_bsp\_syn\_config is shown as below:

**Table 3-133. Function can\_bsp\_syn\_config**

Function name	can_bsp_syn_config
---------------	--------------------

<b>Function prototype</b>	void can_bsp_syn_config(uint32_t can_periph, uint32_t sys_mode);
<b>Function descriptions</b>	configure bit sampling synchronization
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>sys_mode</b>	bit sampling synchronization mode
CAN_BSP_TWO_STAGES_SYN	two stages synchronization for CAN bus sampling
CAN_BSP_ONE_STAGE_SYN	one stage synchronization for CAN bus sampling
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure bit sampling synchronization */
```

```
can_bsp_syn_config(CAN0, CAN_BSP_TWO_STAGES_SYN);
```

## can\_flag\_get

The description of can\_flag\_get is shown as below:

**Table 3-134. Function can\_flag\_get**

<b>Function name</b>	can_flag_get
<b>Function prototype</b>	FlagStatus can_flag_get(uint32_t can_periph, can_flag_enum flag);
<b>Function descriptions</b>	get CAN flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>flag</b>	Refers to enum <a href="#">Table 3-77. Enum can_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-135. Function can\_flag\_clear**

<b>Function name</b>	can_flag_clear
<b>Function prototype</b>	void can_flag_clear(uint32_t can_periph, can_flag_enum flag);
<b>Function descriptions</b>	clear CAN flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>flag</b>	Refers to enum <a href="#">Table 3-77. Enum can_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-136. Function can\_interrupt\_enable**

<b>Function name</b>	can_interrupt_enable
<b>Function prototype</b>	void can_interrupt_enable(uint32_t can_periph, can_interrupt_enum interrupt);
<b>Function descriptions</b>	enable CAN interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	

interrupt	Refers to enum <a href="#">Table 3-76. Enum can_interrupt_enum</a>
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 3-137. 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=0,1,2)	CAN peripheral selection
Input parameter{in}	
interrupt	Refers to enum <a href="#">Table 3-76. Enum can_interrupt_enum</a>
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 3-138. Function can\_interrupt\_flag\_get**

Function name	can_interrupt_flag_get
Function prototype	FlagStatus can_interrupt_flag_get(uint32_t can_periph, can_interrupt_enum int_flag);



Function descriptions	get CAN interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
int_flag	Refers to enum <a href="#">Table 3-78. Enum can_interrupt_flag_enum</a>
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 3-139. 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=0,1,2)	CAN peripheral selection
Input parameter{in}	
int_flag	Refers to enum <a href="#">Table 3-78. Enum can_interrupt_flag_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

## 3.4. CAU

The Cryptographic Acceleration Unit supports acceleration of DES, Triple-DES or AES (128,192, or 256) algorithms. The CAU registers are listed in chapter [3.4.1](#) the CAU firmware functions are introduced in chapter [3.4.2](#)

### 3.4.1. Descriptions of Peripheral registers

CAU registers are listed in the table shown as below:

**Table 3-140. CAU Registers**

Registers	Descriptions
CAU_CTL	control register
CAU_STAT0	status register 0
CAU_DI	data input register
CAU_DO	data output register
CAU_DMAEN	DMA enable register
CAU_INTEN	interrupt enable register
CAU_STAT1	status register 1
CAU_INTF	interrupt flag register
CAU_KEY0H	key 0 high register
CAU_KEY0L	key 0 low register
CAU_KEY1H	key 1 high register
CAU_KEY1L	key 1 low register
CAU_KEY2H	key 2 high register
CAU_KEY2L	key 2 low register
CAU_KEY3H	key 3 high register
CAU_KEY3L	key 3 low register
CAU_IV0H	initial vector 0 high register
CAU_IV0L	initial vector 0 low register
CAU_IV1H	initial vector 1 high register
CAU_IV1L	initial vector 1 low register
CAU_GCMCCMCT XSx (x = 0..7)	GCM or CCM mode context switch register x
CAU_GCMCTXSx (x = 0..7)	GCM mode context switch register x

### 3.4.2. Descriptions of Peripheral functions

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

Table 3-141. CAU firmware function

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

### Structure cau\_key\_parameter\_struct

Table 3-142. Structure cau\_key\_parameter\_struct

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

key_0_high	key 0 high
key_0_low	key 0 low
key_1_high	key 1 high
key_1_low	key 1 low
key_2_high	key 2 high
key_2_low	key 2 low
key_3_high	key 3 high
key_3_low	key 3 low

### Structure cau\_iv\_parameter\_struct

**Table 3-143. Structure cau\_iv\_parameter\_struct**

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

### Structure cau\_context\_parameter\_struct

**Table 3-144. Structure cau\_context\_parameter\_struct**

Member name	Function description
ctl_config	current configuration
iv_0_high	init vector 0 high
iv_0_low	init vector 0 low
iv_1_high	init vector 1 high
iv_1_low	init vector 1 low
key_0_high	key 0 high
key_0_low	key 0 low
key_1_high	key 1 high
key_1_low	key 1 low
key_2_high	key 2 high
key_2_low	key 2 low
key_3_high	key 3 high
key_3_low	key 3 low
gcmccmctxs[8]	GCM or CCM mode context switch
gcmctxs[8]	GCM mode context switch

### Structure cau\_parameter\_struct

**Table 3-145. Structure cau\_parameter\_struct**

Member name	Function description
alg_dir	algorithm directory
*key	key

key_size	key size in bytes
*iv	initialization vector
iv_size	iv size in bytes
*input	input data
in_length	input data length in bytes
*aad	additional authentication data
aad_size	aad size

## cau\_deinit

The description of cau\_deinit is shown as below:

**Table 3-146. Function cau\_deinit**

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

Example:

```
/* reset the CAU peripheral */
cau_deinit();
```

## cau\_struct\_para\_init

The description of cau\_struct\_para\_init is shown as below:

**Table 3-147. Function cau\_struct\_para\_init**

<b>Function name</b>	cau_struct_para_init
<b>Function prototype</b>	void cau_struct_para_init(cau_parameter_struct *cau_parameter);
<b>Function descriptions</b>	initialize the CAU encrypt and decrypt parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145</a> .

	<a href="#"><u>Structure cau_parameter_struct</u></a>
Return value	
-	-

Example:

```
cau_parameter_struct text;
```

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

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

### cau\_key\_struct\_para\_init

The description of cau\_key\_struct\_para\_init is shown as below:

**Table 3-148. Function cau\_key\_struct\_para\_init**

<b>Function name</b>	cau_key_struct_para_init
<b>Function prototype</b>	void cau_key_struct_para_init(cau_key_parameter_struct *key_initpara);
<b>Function descriptions</b>	initialize the key parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>key_initpara</b>	structure for keys initialization of the cau, refer to structure <a href="#"><u>Table 3-142.</u></a> <a href="#"><u>Structure cau_key_parameter_struct</u></a>
<b>Return value</b>	
-	-

Example:

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

```
cau_key_parameter_struct key_initpara;
```

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

### cau\_iv\_struct\_para\_init

The description of cau\_iv\_struct\_para\_init is shown as below:

**Table 3-149. Function cau\_iv\_struct\_para\_init**

<b>Function name</b>	cau_iv_struct_para_init
<b>Function prototype</b>	void cau_iv_struct_para_init(cau_iv_parameter_struct *iv_initpara);
<b>Function descriptions</b>	initialize the vectors parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

-	-
<b>Output parameter{out}</b>	
<b>iv_initpara</b>	structure for vectors initialization of the cau, refer to structure <a href="#">Table 3-143.</a> <a href="#">Structure cau_iv_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize the vectors parameter struct */
```

```
cau_iv_parameter_struct iv_initpara;
```

```
cau_iv_struct_para_init(&iv_initpara);
```

### cau\_context\_struct\_para\_init

The description of cau\_context\_struct\_para\_init is shown as below:

**Table 3-150. Function cau\_context\_struct\_para\_init**

<b>Function name</b>	cau_context_struct_para_init
<b>Function prototype</b>	void cau_context_struct_para_init(cau_context_parameter_struct *cau_context);
<b>Function descriptions</b>	initialize the context parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>cau_context</b>	structure for cau context swapping, refer to structure <a href="#">Table 3-144.</a> <a href="#">Structure cau_context_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

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

```
cau_context_parameter_struct context_initpara;
```

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

### cau\_enable

The description of cau\_enable is shown as below:

**Table 3-151. Function cau\_enable**

<b>Function name</b>	cau_enable
<b>Function prototype</b>	void cau_enable(void);

<b>Function descriptions</b>	enable the CAU peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the CAU peripheral */
```

```
cau_enable();
```

### cau\_disable

The description of cau\_disable is shown as below:

**Table 3-152. Function cau\_disable**

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

Example:

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

```
cau_disable();
```

### cau\_dma\_enable

The description of cau\_dma\_enable is shown as below:

**Table 3-153. Function cau\_dma\_enable**

<b>Function name</b>	cau_dma_enable
<b>Function prototype</b>	void cau_dma_enable(uint32_t dma_req);
<b>Function descriptions</b>	enable the CAU DMA interface



<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_req</b>	specify the CAU DMA transfer request to be enabled
<i>CAU_DMA_INFIFO</i>	DMA for incoming(Rx) data transfer
<i>CAU_DMA_OUTFIFO</i>	DMA for outgoing(Tx) data transfer
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### cau\_dma\_disable

The description of cau\_dma\_disable is shown as below:

**Table 3-154. Function cau\_dma\_disable**

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

Example:

```
/* disable the CAU DMA interface */
cau_dma_disable(CAU_DMA_INFIFO);
```

### cau\_init

The description of cau\_init is shown as below:

Table 3-155. Function cau\_init

<b>Function name</b>	cau_init
<b>Function prototype</b>	void cau_init(uint32_t algo_dir, uint32_t algo_mode, uint32_t swapping);
<b>Function descriptions</b>	initialize the CAU peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>algo_dir</b>	algorithm direction
CAU_ENCRYPT	encrypt
CAU_DECRYPT	decrypt
<b>Input parameter{in}</b>	
<b>algo_mode</b>	algorithm mode selection
CAU_MODE_TDES_ECB	TDES-ECB (3DES Electronic codebook)
CAU_MODE_TDES_CBC	TDES-CBC (3DES Cipher block chaining)
CAU_MODE_DES_ECB	DES-ECB (simple DES Electronic codebook)
CAU_MODE_DES_CBC	DES-CBC (simple DES Cipher block chaining)
CAU_MODE_AES_ECB	AES-ECB (AES Electronic codebook)
CAU_MODE_AES_CBC	AES-CBC (AES Cipher block chaining)
CAU_MODE_AES_CTR	AES-CTR (AES counter mode)
CAU_MODE_AES_KEY	AES decryption key preparation mode
CAU_MODE_AES_GCM	AES-GCM (AES Galois/counter mode)
CAU_MODE_AES_CCM	AES-CCM (AES combined cipher machine mode)
CAU_MODE_AES_CFB	AES-CFB (cipher feedback mode)
CAU_MODE_AES_OFB	AES-OFB (output feedback mode)
<b>Input parameter{in}</b>	
<b>swapping</b>	data swapping selection
CAU_SWAPPING_32BIT	no swapping
CAU_SWAPPING_16BIT	half-word swapping
CAU_SWAPPING_8BIT	bytes swapping

CAU_SWAPPING_1BIT	bit swapping
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the CAU peripheral */
```

```
cau_init(CAU_ENCRYPT, CAU_MODE_TDES_ECB, CAU_SWAPPING_32BIT);
```

### cau\_aes\_keysize\_config

The description of cau\_aes\_keysize\_config is shown as below:

**Table 3-156. Function cau\_aes\_keysize\_config**

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

Example:

```
/* configure key size if used AES algorithm */
```

```
cau_aes_keysize_config(CAU_KEYSIZE_128BIT);
```

### cau\_key\_init

The description of cau\_key\_init is shown as below:

**Table 3-157. Function cau\_key\_init**

Function name	cau_key_init
Function prototype	void cau_key_init(cau_key_parameter_struct* key_initpara);
Function descriptions	initialize the key parameters
Precondition	-

The called functions	-
Input parameter{in}	
key_initpara	structure for keys initialization of the cau, refer to structure <a href="#">Table 3-142.</a> <a href="#">Structure cau_key_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the key parameters */

cau_key_parameter_struct key_initpara;

key_initpara->key_0_high = 0x12345678;
key_initpara->key_0_low = 0x12345678;
key_initpara->key_1_high = 0x12345678;
key_initpara->key_1_low = 0x12345678;
key_initpara->key_2_high = 0x12345678;
key_initpara->key_2_low = 0x12345678;
key_initpara->key_3_high = 0x12345678;
key_initpara->key_4_low = 0x12345678;

cau_key_init(&key_initpara);

```

### cau\_iv\_init

The description of cau\_iv\_init is shown as below:

**Table 3-158. Function cau\_iv\_init**

Function name	cau_iv_init
Function prototype	void cau_iv_init(cau_iv_parameter_struct* iv_initpara);
Function descriptions	initialize the vectors parameters
Precondition	-
The called functions	-
Input parameter{in}	
iv_initpara	structure for vectors initialization of the cau, refer to structure <a href="#">Table 3-143.</a> <a href="#">Structure cau_iv_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the vectors parameters */

cau_iv_parameter_struct iv_initpara;

iv_initpara->iv_0_high = 0x12345678;

iv_initpara->iv_0_low = 0x12345678;

iv_initpara->iv_1_high = 0x12345678;

iv_initpara->iv_1_low = 0x12345678;

cau_iv_init(&iv_initpara);
```

### cau\_phase\_config

The description of cau\_phase\_config is shown as below:

**Table 3-159. Function cau\_phase\_config**

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

Example:

```
/* select prepare phase */

cau_phase_config(CAU_PREPARE_PHASE);
```

### cau\_fifo\_flush

The description of cau\_fifo\_flush is shown as below:

Table 3-160. Function cau\_fifo\_flush

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

Example:

```
/* flush the IN and OUT FIFOs */
cau_fifo_flush();
```

### cau\_enable\_state\_get

The description of cau\_enable\_state\_get is shown as below:

Table 3-161. Function cau\_enable\_state\_get

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

Example:

```
/* return whether CAU peripheral is enabled or disabled */
ControlStatus state = DISABLE;
State = cau_enable_state_get();
```

### cau\_data\_write

The description of cau\_data\_write is shown as below:

Table 3-162. Function cau\_data\_write

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

Example:

```
/* write data to the IN FIFO */
cau_data_write(0x0010);
```

### cau\_data\_read

The description of cau\_data\_read is shown as below:

Table 3-163. Function cau\_data\_read

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

Example:

```
/* return the last data entered into the output FIFO */
uint32_t data;
data = cau_data_read();
```

### cau\_context\_save

The description of cau\_context\_save is shown as below:

Table 3-164. Function cau\_context\_save

Function name	cau_context_save
Function prototype	void cau_context_save(cau_context_parameter_struct *cau_context, cau_key_parameter_struct* key_initpara);
Function descriptions	save context before context switching
Precondition	-
The called functions	-
Input parameter{in}	
key_initpara	structure for keys initialization of the cau, refer to structure <a href="#">Table 3-142. Structure cau_key_parameter_struct</a>
Output parameter{out}	
cau_context	structure for cau context swapping, refer to structure <a href="#">Table 3-144. Structure cau_context_parameter_struct</a>
Return value	
-	-

Example:

```

cau_context_parameter_struct context;

cau_key_parameter_struct key;

cau_parameter_struct cau_parameter;

uint32_t keyaddr;

.....

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

cau_key_struct_para_init(&key);

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

keyaddr += 4U;

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

/* save context before context switching */

cau_context_save(&context, &key);

```

### cau\_context\_restore

The description of cau\_context\_restore is shown as below:

Table 3-165. Function cau\_context\_restore

Function name	cau_context_restore
Function prototype	void cau_context_restore(cau_context_parameter_struct *cau_context);
Function descriptions	restore context after context switching



<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_context</b>	structure for cau context swapping, refer to structure <a href="#">Table 3-144.</a> <a href="#">Structure cau_context_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
cau_context_parameter_struct context;

.....

cau_context_save(&context, &key);

.....

/* restore context after context switching */

cau_context_restore(&context);
```

### cau\_aes\_ecb

The description of cau\_aes\_ecb is shown as below:

**Table 3-166. Function cau\_aes\_ecb**

<b>Function name</b>	cau_aes_ecb
<b>Function prototype</b>	ErrStatus cau_aes_ecb(cau_parameter_struct *cau_parameter, uint8_t *output);
<b>Function descriptions</b>	encrypt and decrypt using AES in ECB mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145.</a> <a href="#">Structure cau_parameter_struct</a>
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[TEXT_SIZE];
```

```

ErrStatus status;

.....

key_addr = key_select[i];

key_size = keysize[i];

text.alg_dir = CAU_ENCRYPT;

text.key = key_addr;

text.key_size = key_size;

text.input = plaintext;

text.in_length = TEXT_SIZE;

/* encryption in ECB mode */

status = cau_aes_ecb(&text, encrypt_result);

```

### cau\_aes\_cbc

The description of cau\_aes\_cbc is shown as below:

**Table 3-167. Function cau\_aes\_cbc**

<b>Function name</b>	cau_aes_cbc
<b>Function prototype</b>	ErrStatus cau_aes_cbc(cau_parameter_struct *cau_parameter, uint8_t *output);
<b>Function descriptions</b>	encrypt and decrypt using AES in CBC mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```

cau_parameter_struct text;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

key_addr = key_select[i];

```

```

key_size = keysize[i];

text.alg_dir = CAU_ENCRYPT;

text.key = key_addr;

text.key_size = key_size;

text.iv = vectors;

text.input = plaintext;

text.in_length = TEXT_SIZE;

/* encryption in CBC mode */

status = cau_aes_cbc(&text, encrypt_result);

```

### cau\_aes\_ctr

The description of cau\_aes\_ctr is shown as below:

**Table 3-168. Function cau\_aes\_ctr**

<b>Function name</b>	cau_aes_ctr
<b>Function prototype</b>	ErrStatus cau_aes_ctr(cau_parameter_struct *cau_parameter, uint8_t *output);
<b>Function descriptions</b>	encrypt and decrypt using AES in CTR mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```

cau_parameter_struct text;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

key_addr = key_select[i];

key_size = keysize[i];

text.alg_dir = CAU_ENCRYPT;

```

```

text.key = key_addr;

text.key_size = key_size;

text.iv = vectors;

text.input = plaintext;

text.in_length = TEXT_SIZE;

/* encryption in CTR mode */

status = cau_aes_ctr(&text, encrypt_result);

```

### cau\_aes\_cfb

The description of cau\_aes\_cfb is shown as below:

**Table 3-169. Function cau\_aes\_cfb**

<b>Function name</b>	cau_aes_cfb
<b>Function prototype</b>	ErrStatus cau_aes_cfb(cau_parameter_struct *cau_parameter, uint8_t *output);
<b>Function descriptions</b>	encrypt and decrypt using AES in CFB mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```

cau_parameter_struct cau_cfb_parameter;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

/* encryption in CFB mode */

cau_cfb_parameter.alg_dir = CAU_ENCRYPT;

cau_cfb_parameter.key = (uint8_t *)key_128;

cau_cfb_parameter.key_size = KEY_SIZE;

cau_cfb_parameter.iv = (uint8_t *)vectors;

```

```

cau_cfb_parameter.iv_size = IV_SIZE;

cau_cfb_parameter.input = (uint8_t *)plaintext;

cau_cfb_parameter.in_length = PLAINTEXT_SIZE;

status = cau_aes_cfb(&cau_cfb_parameter, encrypt_result);

```

### cau\_aes\_ofb

The description of cau\_aes\_ofb is shown as below:

**Table 3-170. Function cau\_aes\_ofb**

<b>Function name</b>	cau_aes_ofb
<b>Function prototype</b>	ErrStatus cau_aes_ofb(cau_parameter_struct *cau_parameter, uint8_t *output);
<b>Function descriptions</b>	encrypt and decrypt using AES in OFB mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```

cau_parameter_struct cau_ofb_parameter;

uint8_t encrypt_result[TEXT_SIZE];

ErrStatus status;

.....

/* encryption in OFB mode */

cau_ofb_parameter.alg_dir = CAU_ENCRYPT;

cau_ofb_parameter.key = (uint8_t *)key_128;

cau_ofb_parameter.key_size = KEY_SIZE;

cau_ofb_parameter.iv = (uint8_t *)vectors;

cau_ofb_parameter.iv_size = IV_SIZE;

cau_ofb_parameter.input = (uint8_t *)plaintext;

cau_ofb_parameter.in_length = PLAINTEXT_SIZE;

```

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

## cau\_aes\_gcm

The description of cau\_aes\_gcm is shown as below:

**Table 3-171. Function cau\_aes\_gcm**

<b>Function name</b>	cau_aes_gcm
<b>Function prototype</b>	ErrStatus cau_aes_gcm(cau_parameter_struct *cau_parameter, uint8_t *output, uint8_t *tag);
<b>Function descriptions</b>	encrypt and decrypt using AES in GCM mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Output parameter{out}</b>	
<b>tag</b>	pointer to the returned tag buffer
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```
cau_parameter_struct cau_gcm_parameter;
```

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

```
uint8_t gcm_tag[GCM_TAG_SIZE];
```

```
ErrStatus status;
```

```
.....
```

```
/* encryption in GCM mode */
```

```
cau_gcm_parameter.alg_dir = CAU_ENCRYPT;
```

```
cau_gcm_parameter.key = (uint8_t *)key_128;
```

```
cau_gcm_parameter.key_size = KEY_SIZE;
```

```
cau_gcm_parameter.iv = (uint8_t *)vectors;
```

```
cau_gcm_parameter.iv_size = IV_SIZE;
```

```
cau_gcm_parameter.input = (uint8_t *)plaintext;
```

```
cau_gcm_parameter.in_length = PLAINTEXT_SIZE;
```

```
cau_gcm_parameter.aad = (uint8_t *)aadmessage;
```

```
cau_gcm_parameter.aad_size = AAD_SIZE;
```

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

### cau\_aes\_ccm

The description of cau\_aes\_ccm is shown as below:

**Table 3-172. Function cau\_aes\_ccm**

<b>Function name</b>	cau_aes_ccm
<b>Function prototype</b>	ErrStatus cau_aes_ccm(cau_parameter_struct *cau_parameter, uint8_t *output, uint8_t tag[], uint32_t tag_size, uint8_t aad_buf[]);
<b>Function descriptions</b>	encrypt and decrypt using AES in CCM mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
<b>Input parameter{in}</b>	
<b>tag_size</b>	tag size (in bytes)
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Output parameter{out}</b>	
<b>tag</b>	pointer to the returned tag buffer
<b>Output parameter{out}</b>	
<b>aad_buf</b>	pointer to the user buffer used when formatting aad block
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```
cau_parameter_struct cau_ccm_parameter;

uint8_t encrypt_result[TEXT_SIZE];

uint8_t ccm_tag[CCM_TAG_SIZE];

uint8_t aad_buf[AAD_SIZE + 21];

ErrStatus status;

.....

/* encryption in CCM mode */

cau_ccm_parameter.alg_dir = CAU_ENCRYPT;

cau_ccm_parameter.key = (uint8_t *)ccm_key_128;

cau_ccm_parameter.key_size = KEY_SIZE;
```

```

cau_ccm_parameter.iv = (uint8_t *)ccm_vectors;

cau_ccm_parameter.iv_size = CCM_IV_SIZE;

cau_ccm_parameter.input = (uint8_t *)plaintext;

cau_ccm_parameter.in_length = PLAINTEXT_SIZE;

cau_ccm_parameter.aad = (uint8_t *)aadmessage;

cau_ccm_parameter.aad_size = AAD_SIZE;

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

```

### cau\_tdes\_ecb

The description of cau\_tdes\_ecb is shown as below:

**Table 3-173. Function cau\_tdes\_ecb**

Function name	cau_tdes_ecb
Function prototype	ErrStatus cau_tdes_ecb(cau_parameter_struct *cau_parameter, uint8_t *output);
Function descriptions	encrypt and decrypt using TDES in ECB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```

cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

text.alg_dir = CAU_ENCRYPT;

text.key = tdes_key;

text.input = plaintext;

text.in_length = DATA_SIZE;

```



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

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

### cau\_tdes\_cbc

The description of cau\_tdes\_cbc is shown as below:

**Table 3-174. Function cau\_tdes\_cbc**

<b>Function name</b>	cau_tdes_cbc
<b>Function prototype</b>	ErrStatus cau_tdes_cbc(cau_parameter_struct *cau_parameter, uint8_t *output);
<b>Function descriptions</b>	encrypt and decrypt using TDES in CBC mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;
```

```
uint8_t encrypt_result[DATA_SIZE];
```

```
ErrStatus status;
```

```
.....
```

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

```
text.key = tdes_key;
```

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

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

```
text.in_length = DATA_SIZE;
```

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

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

### cau\_des\_ecb

The description of cau\_des\_ecb is shown as below:

Table 3-175. Function cau\_des\_ecb

Function name	cau_des_ecb
Function prototype	ErrStatus cau_des_ecb(cau_parameter_struct *cau_parameter, uint8_t *output);
Function descriptions	encrypt and decrypt using DES in ECB mode
Precondition	-
The called functions	-
Input parameter{in}	
cau_parameter	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
Output parameter{out}	
output	pointer to the returned buffer
Return value	
ErrStatus	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

text.alg_dir = CAU_ENCRYPT;

text.key = des_key;

text.input = plaintext;

text.in_length = DATA_SIZE;

/* encryption in ECB mode */

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

### cau\_des\_cbc

The description of cau\_des\_cbc is shown as below:

Table 3-176. Function cau\_des\_cbc

Function name	cau_des_cbc
Function prototype	ErrStatus cau_des_cbc(cau_parameter_struct *cau_parameter, uint8_t *output);
Function descriptions	encrypt and decrypt using DES in CBC mode
Precondition	-
The called functions	-
Input parameter{in}	

<b>cau_parameter</b>	structure for encrypt and decrypt parameters, refer to structure <a href="#">Table 3-145. Structure cau_parameter_struct</a>
<b>Output parameter{out}</b>	
<b>output</b>	pointer to the returned buffer
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```
cau_parameter_struct text;

uint8_t encrypt_result[DATA_SIZE];

ErrStatus status;

.....

text.alg_dir = CAU_ENCRYPT;

text.key = des_key;

text.iv = vectors;

text.input = plaintext;

text.in_length = DATA_SIZE;

/* encryption in CBC mode */

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

## cau\_flag\_get

The description of cau\_flag\_get is shown as below:

**Table 3-177. Function cau\_flag\_get**

<b>Function name</b>	cau_flag_get
<b>Function prototype</b>	FlagStatus cau_flag_get(uint32_t flag);
<b>Function descriptions</b>	get the CAU flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	CAU flag status
CAU_FLAG_INFIFO_EMPTY	input FIFO empty
CAU_FLAG_INFIFO_NOT_FULL	input FIFO is not full
CAU_FLAG_OUTFIFO_NOT_EMPTY	output FIFO not empty
CAU_FLAG_OUTFIFO_FULL	output FIFO is full

<i>_FULL</i>	
<i>CAU_FLAG_BUSY</i>	the CAU core is busy
<i>CAU_FLAG_INFIFO</i>	input FIFO flag status
<i>CAU_FLAG_OUTFIFO</i>	output FIFO flag status
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

```
FlagStatus status = RESET;
```

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

### cau\_interrupt\_enable

The description of cau\_interrupt\_enable is shown as below:

**Table 3-178. Function cau\_interrupt\_enable**

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

Example:

```
/* enable cau interrupt */
```

```
cau_interrupt_enable(CAU_INT_INFIFO);
```

### cau\_interrupt\_disable

The description of cau\_interrupt\_disable is shown as below:

**Table 3-179. Function cau\_interrupt\_disable**

<b>Function name</b>	cau_interrupt_disable
----------------------	-----------------------

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

Example:

```
/* disable cau interrupt */
cau_interrupt_disable(CAU_INT_INFIFO);
```

### cau\_interrupt\_flag\_get

The description of cau\_interrupt\_flag\_get is shown as below:

**Table 3-180. Function cau\_interrupt\_flag\_get**

<b>Function name</b>	cau_interrupt_flag_get
<b>Function prototype</b>	FlagStatus cau_interrupt_flag_get(uint32_t int_flag);
<b>Function descriptions</b>	get the interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	CAU interrupt flag
CAU_INT_FLAG_INFIFO	input FIFO interrupt
CAU_INT_FLAG_OUTFIFO	output FIFO interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the CAU interrupt flag status */
FlagStatus status = RESET;
status = cau_interrupt_flag_get(CAU_INT_FLAG_INFIFO);
```

## 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 3-181. CMP registers**

Registers	Descriptions
CMP_STAT	CMP status register
CMP_IFC	CMP interrupt flag clear register
CMP_SR	CMP alternate select register
CMP0_CS	CMP0 control and status register
CMP1_CS	CMP1 control and status register

### 3.5.2. Descriptions of Peripheral functions

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

**Table 3-182. CMP firmware function**

Function name	Function description
cmp_deinit	CMP deinit
cmp_mode_init	CMP mode init
cmp_noninverting_input_select	CMP noninverting input select
cmp_output_init	CMP output init
cmp_output_mux_config	config comparator output port
cmp_blanking_init	CMP output blanking function init
cmp_enable	enable CMP
cmp_disable	disable CMP
cmp_window_enable	enable the window mode
cmp_window_disable	disable the window mode
cmp_lock_enable	lock the CMP
cmp_voltage_scaler_enable	enable the voltage scaler
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_output_level_get	get output level
cmp_flag_get	get CMP flag

Function name	Function description
cmp_flag_clear	clear CMP flag
cmp_interrupt_enable	enable CMP interrupt
cmp_interrupt_disable	disable CMP interrupt
cmp_interrupt_flag_get	get CMP interrupt flag
cmp_interrupt_flag_clear	clear CMP interrupt flag

## Enum cmp\_enum

**Table 3-183. Enum cmp\_enum**

Member name	Function description
CMP0	comparator 0
CMP1	comparator 1

## cmp\_deinit

The description of cmp\_deinit is shown as below:

**Table 3-184. Function cmp\_deinit**

Function name	cmp_deinit
Function prototype	void cmp_deinit(cmp_enum cmp_periph);
Function descriptions	CMP deinit
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize CMP0 */
cmp_deinit(CMP0);
```

## cmp\_mode\_init

The description of cmp\_mode\_init is shown as below:

**Table 3-185. Function cmp\_mode\_init**

Function name	cmp_mode_init
Function prototype	void cmp_mode_init(cmp_enum cmp_periph, uint32_t operating_mode, uint32_t inverting_input, uint32_t output_hysteresis);
Function descriptions	CMP mode init

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>operating_mode</b>	operating mode
<b>CMP_MODE_HIGHSP EED</b>	high speed mode
<b>CMP_MODE_MIDDLE SPEED</b>	medium speed mode
<b>CMP_MODE_VERYLO WSPEED</b>	very-low speed mode
<b>Input parameter{in}</b>	
<b>inverting_input</b>	inverting input
<b>CMP_INVERTING_INP UT_1_4VREFINT</b>	VREFINT *1/4 input
<b>CMP_INVERTING_INP UT_1_2VREFINT</b>	VREFINT *1/2 input
<b>CMP_INVERTING_INP UT_3_4VREFINT</b>	VREFINT *3/4 input
<b>CMP_INVERTING_INP UT_VREFINT</b>	VREFINT input
<b>CMP_INVERTING_INP UT_DAC0_OUT0</b>	PA4 (DAC) input
<b>CMP_INVERTING_INP UT_DAC0_OUT1</b>	PA5 (DAC) input
<b>CMP_INVERTING_INP UT_PB1_PE10</b>	PB1 for CMP0 or PE10 for CMP1 as inverting input
<b>CMP_INVERTING_INP UT_PC4_PE7</b>	PC4 for CMP0 or PE7 for CMP1 as inverting input
<b>Input parameter{in}</b>	
<b>output_hysteresis</b>	hysteresis level
<b>CMP_HYSTERESIS_N O</b>	output no hysteresis
<b>CMP_HYSTERESIS_L OW</b>	output low hysteresis
<b>CMP_HYSTERESIS_M IDDLE</b>	output middle hysteresis
<b>CMP_HYSTERESIS_HI GH</b>	output high hysteresis
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	



-	-
---	---

Example:

```
/* initialize CMP0 mode */
```

```
cmp_mode_init(CMP0, CMP_MODE_HIGHSPEED, CMP_1_4VREFINT, CMP_HYSTERE
SIS_NO);
```

### cmp\_noninverting\_input\_select

The description of cmp\_noninverting\_input\_select is shown as below:

**Table 3-186. Function cmp\_noninverting\_input\_select**

<b>Function name</b>	cmp_noninverting_input_select
<b>Function prototype</b>	void cmp_noninverting_input_select(cmp_enum cmp_periph, uint32_t noninverting_input);
<b>Function descriptions</b>	CMP noninverting input select
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>noninverting_input</b>	noninverting input select
<b>CMP_NONINVERTING_INPUT_PB0_PE9</b>	CMP noninverting input PB0 for CMP0 or PE9 for CMP1
<b>CMP_NONINVERTING_INPUT_PB2_PE12</b>	CMP noninverting input PB2 for CMP0 or PE12 for CMP1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* selecte the noninverting input for CMP0 */
```

```
cmp_noninverting_input_select(CMP0, CMP_NONINVERTING_INPUT_PB0_PE9);
```

### cmp\_output\_init

The description of cmp\_output\_init is shown as below:

**Table 3-187. Function cmp\_output\_init**

<b>Function name</b>	cmp_output_init
<b>Function prototype</b>	void cmp_output_init(cmp_enum cmp_periph, uint32_t output_polarity);
<b>Function descriptions</b>	CMP output init
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Input parameter{in}	
output_polarity	CMP output polarity
CMP_OUTPUT_POLARITY_INVERTED	output is inverted
CMP_OUTPUT_POLARITY_NONINVERTED	output is not inverted
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize CMP0 output */
```

```
cmp_output_init(CMP0, CMP_OUTPUT_POLARITY_NOINVERTED);
```

### cmp\_output\_mux\_config

The description of cmp\_output\_mux\_config is shown as below:

**Table 3-188. Function cmp\_output\_mux\_config**

Function name	cmp_output_mux_config
Function prototype	void cmp_output_mux_config(cmp_enum cmp_periph, uint32_t cmp_output_sel);
Function descriptions	config comparator output port
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Input parameter{in}	
cmp_output_sel	CMP output selection
CMP_AFSE_GPIO_PA6	CMP alternate GPIO PA6
CMP_AFSE_GPIO_PA8	CMP alternate GPIO PA8
CMP_AFSE_GPIO_PB12	CMP alternate GPIO PB12
CMP_AFSE_GPIO_PE6	CMP alternate GPIO PE6
CMP_AFSE_GPIO_PE15	CMP alternate GPIO PE15

<i>CMP_AFSE_GPIO_PG</i> 2	CMP alternate GPIO PG2
<i>CMP_AFSE_GPIO_PG</i> 3	CMP alternate GPIO PG3
<i>CMP_AFSE_GPIO_PG</i> 4	CMP alternate GPIO PG4
<i>CMP_AFSE_GPIO_PK</i> 0	CMP alternate GPIO PK0
<i>CMP_AFSE_GPIO_PK</i> 1	CMP alternate GPIO PK1
<i>CMP_AFSE_GPIO_PK</i> 2	CMP alternate GPIO PK2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config CMP0 output port */
```

```
cmp_output_mux_config(CMP0, CMP_AFSE_GPIO_PA6);
```

### cmp\_blanking\_init

The description of cmp\_blanking\_init is shown as below:

**Table 3-189. Function cmp\_outputblank\_init**

<b>Function name</b>	cmp_blanking_init
<b>Function prototype</b>	void cmp_blanking_init(cmp_enum cmp_periph, uint32_t blanking_source_selection);
<b>Function descriptions</b>	CMP output blanking function init
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>blanking_source_selection</b>	blanking source selection
<i>CMP_BLANKING_NONE</i>	CMP no blanking source
<i>CMP_BLANKING_TIMER0_OC0</i>	CMP TIMER0_CH0 output compare signal selected as blanking source
<i>CMP_BLANKING_TIMER1_OC2</i>	CMP TIMER1_CH2 output compare signal selected as blanking source

<i>CMP_BLANKING_TIM</i> <i>ER2_OC2</i>	CMP TIMER2_CH2 output compare signal selected as blanking source
<i>CMP_BLANKING_TIM</i> <i>ER2_OC3</i>	CMP TIMER2_CH3 output compare signal selected as blanking source
<i>CMP_BLANKING_TIM</i> <i>ER7_OC0</i>	CMP TIMER7_CH0 output compare signal selected as blanking source
<i>CMP_BLANKING_TIM</i> <i>ER14_OC0</i>	CMP TIMER14_CH0 output compare signal selected as blanking source
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize CMP0 blanking function */
cmp_blanking_init(CMP0, CMP_BLANKING_NONE);
```

### cmp\_enable

The description of cmp\_enable is shown as below:

**Table 3-190. Function cmp\_enable**

<b>Function name</b>	cmp_enable
<b>Function prototype</b>	void cmp_enable(cmp_enum cmp_periph);
<b>Function descriptions</b>	enable CMP
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CMP0 */
cmp_enable(CMP0);
```

### cmp\_disable

The description of cmp\_disable is shown as below:

**Table 3-191. Function cmp\_disable**

<b>Function name</b>	cmp_disable
----------------------	-------------

Function prototype	void cmp_disable(cmp_enum cmp_periph);
Function descriptions	disable CMP
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CMP0 */
cmp_disable(CMP0);
```

### cmp\_window\_enable

The description of cmp\_window\_enable is shown as below:

**Table 3-192. Function cmp\_window\_enable**

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

Example:

```
/* enable the window mode */
cmp_window_enable();
```

### cmp\_window\_disable

The description of cmp\_window\_disable is shown as below:

**Table 3-193. Function cmp\_window\_disable**

Function name	cmp_window_disable
Function prototype	void cmp_window_disable(void);

<b>Function descriptions</b>	disable the window mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the window mode */
```

```
cmp_window_disable();
```

### cmp\_lock\_enable

The description of cmp\_lock\_enable is shown as below:

**Table 3-194. Function cmp\_lock\_enable**

<b>Function name</b>	cmp_lock_enable
<b>Function prototype</b>	void cmp_lock_enable(cmp_enum cmp_periph);
<b>Function descriptions</b>	lock the comparator
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* lock CMP0 register */
```

```
cmp_lock_enable(CMP0);
```

### cmp\_voltage\_scaler\_enable

The description of cmp\_voltage\_scaler\_enable is shown as below:

**Table 3-195. Function cmp\_voltage\_scaler\_enable**

<b>Function name</b>	cmp_voltage_scaler_enable
<b>Function prototype</b>	void cmp_voltage_scaler_enable(cmp_enum cmp_periph);
<b>Function descriptions</b>	enable the voltage scaler

Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CMP0 the voltage scaler */
cmp_voltage_scaler_enable(CMP0);
```

### cmp\_voltage\_scaler\_disable

The description of cmp\_voltage\_scaler\_disable is shown as below:

**Table 3-196. Function cmp\_voltage\_scaler\_disable**

Function name	cmp_voltage_scaler_disable
Function prototype	void cmp_voltage_scaler_disable(cmp_enum cmp_periph);
Function descriptions	disable comparator the voltage scaler
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CMP0 the voltage scaler */
cmp_voltage_scaler_disable(CMP0);
```

### cmp\_scaler\_bridge\_enable

The description of cmp\_scaler\_bridge\_enable is shown as below:

**Table 3-197. Function cmp\_scaler\_bridge\_enable**

Function name	cmp_scaler_bridge_enable
Function prototype	void cmp_scaler_bridge_enable(cmp_enum cmp_periph);
Function descriptions	enable the scaler bridge
Precondition	-

The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CMP0 the scaler bridge */
cmp_scaler_bridge_enable(CMP0);
```

### cmp\_scaler\_bridge\_disable

The description of cmp\_scaler\_bridge\_disable is shown as below:

**Table 3-198. Function cmp\_scaler\_bridge\_disable**

Function name	cmp_scaler_bridge_disable
Function prototype	void cmp_scaler_bridge_disable(cmp_enum cmp_periph);
Function descriptions	disable the scaler bridge
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CMP0 the scaler bridge */
cmp_scaler_bridge_disable(CMP0);
```

### cmp\_output\_level\_get

The description of cmp\_output\_level\_get is shown as below:

**Table 3-199. Function cmp\_output\_level\_get**

Function name	cmp_output_level_get
Function prototype	uint32_t cmp_output_level_get(uint32_t cmp_periph);
Function descriptions	get output level
Precondition	-
The called functions	-



Input parameter{in}	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
<b>uint32_t</b>	the output level
<b>CMP_OUTPUTLEVEL_HIGH</b>	comparator output high
<b>CMP_OUTPUTLEVEL_LOW</b>	comparator output low

Example:

```
uint32_t level;
```

```
/* get CMP0 output level */
```

```
level = cmp_output_level_get(CMP0);
```

### cmp\_flag\_get

The description of cmp\_flag\_get is shown as below:

**Table 3-200. Function cmp\_flag\_get**

<b>Function name</b>	cmp_flag_get
<b>Function prototype</b>	FlagStatus cmp_flag_get(cmp_enum cmp_periph, uint32_t flag);
<b>Function descriptions</b>	get CMP flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Input parameter{in}	
<b>flag</b>	CMP interrupt flag
<b>CMP_FLAG_COMPARE</b>	CMP compare flag
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the CMP0 interrupt flag bit */
```

```
FlagStatus flag_value;
```

```
flag_value = cmp_flag_get(CMP0, CMP_FLAG_COMPARE);
```

## cmp\_flag\_clear

The description of cmp\_flag\_clear is shown as below:

**Table 3-201. Function cmp\_flag\_clear**

<b>Function name</b>	cmp_flag_clear
<b>Function prototype</b>	void cmp_flag_clear(cmp_enum cmp_periph, uint32_t flag);
<b>Function descriptions</b>	clear CMP flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	CMP interrupt flag
<b>CMP_FLAG_COMPAR E</b>	CMP compare flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the CMP0 interrupt flag bit */
```

```
cmp_flag_clear(CMP0, CMP_FLAG_COMPARE);
```

## cmp\_interrupt\_enable

The description of cmp\_interrupt\_enable is shown as below:

**Table 3-202. Function cmp\_interrupt\_enable**

<b>Function name</b>	cmp_interrupt_enable
<b>Function prototype</b>	void cmp_interrupt_enable(cmp_enum cmp_periph, uint32_t interrupt);
<b>Function descriptions</b>	enable CMP interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>interrupt</b>	CMP interrupt
<b>CMP_INT_COMPARE</b>	CMP compare interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the CMP0 interrupt */

cmp_interrupt_enable(CMP0, CMP_INT_COMPARE);
```

### cmp\_interrupt\_disable

The description of cmp\_interrupt\_disable is shown as below:

**Table 3-203. Function cmp\_interrupt\_disable**

<b>Function name</b>	cmp_interrupt_disable
<b>Function prototype</b>	void cmp_interrupt_disable(cmp_enum cmp_periph, uint32_t interrupt);
<b>Function descriptions</b>	disable CMP interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>interrupt</b>	CMP interrupt
<b>CMP_INT_COMPARE</b>	CMP compare interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the CMP0 interrupt */

cmp_interrupt_disable(CMP0, CMP_INT_COMPARE);
```

### cmp\_interrupt\_flag\_get

The description of cmp\_interrupt\_flag\_get is shown as below:

**Table 3-204. Function cmp\_interrupt\_flag\_get**

<b>Function name</b>	cmp_interrupt_flag_get
<b>Function prototype</b>	FlagStatus cmp_interrupt_flag_get(cmp_enum cmp_periph, uint32_t flag);
<b>Function descriptions</b>	get CMP interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	CMP interrupt flag
<b>CMP_INT_FLAG_COM</b>	CMP compare interrupt flag

<i>PARE</i>	
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the CMP0 interrupt bit*/
```

```
FlagStatus flag_value;
```

```
flag_value = cmp_interrupt_flag_get(CMP0, CMP_INT_FLAG_COMPARE);
```

### cmp\_interrupt\_flag\_clear

The description of cmp\_interrupt\_flag\_clear is shown as below:

**Table 3-205. Function cmp\_interrupt\_flag\_clear**

Function name	cmp_interrupt_flag_clear
Function prototype	void cmp_interrupt_flag_clear(cmp_enum cmp_periph, uint32_t flag);
Function descriptions	clear CMP interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-183. Enum cmp_enum</a>
Input parameter{in}	
flag	CMP interrupt flag
CMP_INT_FLAG_COMPARE	CMP compare interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the CMP0 interrupt bit*/
```

```
cmp_interrupt_flag_clear(CMP0, CMP_INT_FLAG_COMPARE );
```

## 3.6. CPDM

The Clock Phase Delay Module (CPDM) is used to delay the phase of the input clock and then output the clock. The CPDM registers are listed in chapter [3.6.1](#), the CPDM firmware functions are introduced in chapter [3.6.2](#).

### 3.6.1. Descriptions of Peripheral registers

CPDM registers are listed in the table shown as below:

**Table 3-206. CPDM Registers**

Registers	Descriptions
CPDM_CTL	CPDM control register
CPDM_CFG	CPDM configuration register

### 3.6.2. Descriptions of Peripheral functions

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

**Table 3-207. CPDM firmware function**

Function name	Function description
cpdm_enable	enable CPDM
cpdm_disable	disable CPDM
cpdm_delayline_sample_enable	enable CPDM delay line sample module
cpdm_delayline_sample_disable	disable CPDM delay line sample module
cpdm_output_clock_phase_select	select CPDM output clock phase
cpdm_delay_step_config	configure CPDM delay step
cpdm_delayline_length_valid_flag_get	get delay line length valid flag
cpdm_delayline_length_get	get delay line length
cpdm_clock_output	configure CPDM clock output

#### Enum cpdm\_output\_phase\_enum

**Table 3-208. Enum cpdm\_output\_phase\_enum**

Member name	Function description
CPDM_OUTPUT_PHASE_SELECTION_0	output clock phase = input clock
CPDM_OUTPUT_PHASE_SELECTION_1	output clock phase = input clock + 1 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_2	output clock phase = input clock + 2 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_3	output clock phase = input clock + 3 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_4	output clock phase = input clock + 4 * UNIT delay

Member name	Function description
CPDM_OUTPUT_P HASE_SELECTION _5	output clock phase = input clock + 5 * UNIT delay
CPDM_OUTPUT_P HASE_SELECTION _6	output clock phase = input clock + 6 * UNIT delay
CPDM_OUTPUT_P HASE_SELECTION _7	output clock phase = input clock + 7 * UNIT delay
CPDM_OUTPUT_P HASE_SELECTION _8	output clock phase = input clock + 8 * UNIT delay
CPDM_OUTPUT_P HASE_SELECTION _9	output clock phase = input clock + 9 * UNIT delay
CPDM_OUTPUT_P HASE_SELECTION _10	output clock phase = input clock + 10 * UNIT delay
CPDM_OUTPUT_P HASE_SELECTION _11	output clock phase = input clock + 11 * UNIT delay
CPDM_OUTPUT_P HASE_SELECTION _12	output clock phase = input clock + 12 * UNIT delay

### cpdm\_enable

The description of cpdm\_enable is shown as below:

**Table 3-209. Function cpdm\_enable**

<b>Function name</b>	cpdm_enable
<b>Function prototype</b>	void cpdm_enable(void);
<b>Function descriptions</b>	enable CPDM
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CPDM */
```

```
cpdm_enable();
```

### cpdm\_disable

The description of cpdm\_disable is shown as below:

**Table 3-210. Function cpdm\_disable**

<b>Function name</b>	cpdm_disable
<b>Function prototype</b>	void cpdm_disable(void);
<b>Function descriptions</b>	disable CPDM
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CPDM */
```

```
cpdm_disable();
```

### cpdm\_delayline\_sample\_enable

The description of cpdm\_delayline\_sample\_enable is shown as below:

**Table 3-211. Function cpdm\_delayline\_sample\_enable**

<b>Function name</b>	cpdm_delayline_sample_enable
<b>Function prototype</b>	void cpdm_delayline_sample_enable(void);
<b>Function descriptions</b>	enable CPDM delay line sample module
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CPDM delay line sample module */
```

```
cpdm_delayline_sample_enable();
```

### cpdm\_delayline\_sample\_disable

The description of cpdm\_delayline\_sample\_disable is shown as below:

**Table 3-212. Function cpdm\_delayline\_sample\_disable**

<b>Function name</b>	cpdm_delayline_sample_disable
<b>Function prototype</b>	void cpdm_delayline_sample_disable(void);
<b>Function descriptions</b>	disable CPDM delay line sample module
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CPDM delay line sample module */
```

```
cpdm_delayline_sample_disable();
```

### cpdm\_output\_clock\_phase\_select

The description of cpdm\_output\_clock\_phase\_select is shown as below:

**Table 3-213. Function cpdm\_output\_clock\_phase\_select**

<b>Function name</b>	cpdm_output_clock_phase_select
<b>Function prototype</b>	void cpdm_output_clock_phase_select(cpdm_output_phase_enum output_clock_phase);
<b>Function descriptions</b>	select CPDM output clock phase
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
output_clock_phase	the output clock phase, refer to <a href="#">Table 3-208. Enum cpdm_output_phase_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select CPDM output clock phase */
```



```
cpdm_output_clock_phase_select(CPDM_OUTPUT_PHASE_SELECTION_0);
```

## cpdm\_delay\_step\_config

The description of cpdm\_delay\_step\_config is shown as below:

**Table 3-214. Function cpdm\_delay\_step\_config**

<b>Function name</b>	cpdm_delay_step_config
<b>Function prototype</b>	void cpdm_delay_step_config(uint8_t delay_step)
<b>Function descriptions</b>	configure CPDM delay step
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>delay_step</b>	0 ~ 127
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CPDM delay step */
cpdm_delay_step_config(1);
```

## cpdm\_delayline\_length\_valid\_flag\_get

The description of cpdm\_delayline\_length\_valid\_flag\_get is shown as below:

**Table 3-215. Function cpdm\_delayline\_length\_valid\_flag\_get**

<b>Function name</b>	cpdm_delayline_length_valid_flag_get
<b>Function prototype</b>	FlagStatus cpdm_delayline_length_valid_flag_get(void);
<b>Function descriptions</b>	get delay line length valid flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get delay line length valid flag */
FlagStatus flag;
```

```
flag = cpdm_delayline_length_valid_flag_get();
```

## cpdm\_delayline\_length\_get

The description of cpdm\_delayline\_length\_get is shown as below:

**Table 3-216. Function cpdm\_delayline\_length\_get**

<b>Function name</b>	cpdm_delayline_length_get
<b>Function prototype</b>	uint16_t cpdm_delayline_length_get(void);
<b>Function descriptions</b>	get delay line length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint16_t	0x00~0xFF

Example:

```
/* get delay line length */
uint16_t len;
len = cpdm_delayline_length_get();
```

## cpdm\_clock\_output

The description of cpdm\_clock\_output is shown as below:

**Table 3-217. Function cpdm\_clock\_output**

<b>Function name</b>	cpdm_clock_output
<b>Function prototype</b>	void cpdm_clock_output(cpdm_output_phase_enum output_clock_phase);
<b>Function descriptions</b>	configure CPDM clock output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
output_clock_phase	the output clock phase, refer to <a href="#">Table 3-208. Enum cpdm_output_phase_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CPDM clock output */
```

```
cpdm_clock_output(CPDM_OUTPUT_PHASE_SELECTION_1);
```

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

### 3.7.1. Descriptions of Peripheral registers

CRC registers are listed in the table shown as below:

**Table 3-218. 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

### 3.7.2. Descriptions of Peripheral functions

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

**Table 3-219. 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 initial 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 3-220. Function `crc_deinit`

Function name	<code>crc_deinit</code>
Function prototype	<code>void crc_deinit(void);</code>
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 3-221. Function `crc_reverse_output_data_enable`

Function name	<code>crc_reverse_output_data_enable</code>
Function prototype	<code>void crc_reverse_output_data_enable(void);</code>
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 3-222. Function `crc_reverse_output_data_disable`

Function name	<code>crc_reverse_output_data_disable</code>
Function prototype	<code>void crc_reverse_output_data_disable(void);</code>
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 3-223. Function `crc_data_register_reset`

Function name	<code>crc_data_register_reset</code>
Function prototype	<code>void crc_data_register_reset(void);</code>
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 3-224. Function `crc_data_register_read`

Function name	<code>crc_data_register_read</code>
Function prototype	<code>uint32_t crc_data_register_read(void);</code>
Function descriptions	read the data register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>uint32_t</code>	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 3-225. 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	
<code>uint8_t</code>	8-bit value of the free data register (0-0xFF)

Example:

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

uint8_t crc_value = 0;

crc_value = crc_free_data_register_read();
```

### `crc_free_data_register_write`

The description of `crc_free_data_register_write` is shown as below:

Table 3-226. 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}	
<code>free_data</code>	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 3-227. 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}	
<code>init_data</code>	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 3-228. Function `crc_input_data_reverse_config`

<b>Function name</b>	<code>crc_input_data_reverse_config</code>
<b>Function prototype</b>	<code>void crc_input_data_reverse_config(uint32_t data_reverse)</code>
<b>Function descriptions</b>	configure the crc input data function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data_reverse</b>	specify input data reverse function
<code>CRC_INPUT_DATA_NOT</code>	input data is not reversed
<code>CRC_INPUT_DATA_BYTE</code>	input data is reversed on 8 bits
<code>CRC_INPUT_DATA_HALFWORD</code>	input data is reversed on 16 bits
<code>CRC_INPUT_DATA_WORD</code>	input data is reversed on 32 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-229. Function `crc_polynomial_size_set`

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



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 3-230. Function crc\_polynomial\_set**

Function name	crc_polynomial_set
Function prototype	void crc_polynomial_set(uint32_t poly)
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 3-231. Function crc\_single\_data\_calculate**

Function name	crc_single_data_calculate
Function prototype	uint32_t crc_single_data_calculate(uint32_t sdata, uint8_t data_format);
Function descriptions	CRC calculate single data
Precondition	-
The called functions	-
Input parameter{in}	
sdata	specify 32-bit data
Input parameter{in}	
data_format	input data format
INPUT_FORMAT_WO	input data in word format

<i>RD</i>	
<i>INPUT_FORMAT_HAL FWORD</i>	input data in half-word format
<i>INPUT_FORMAT_BYT E</i>	input data in byte format
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	32-bit 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 3-232. Function `crc_block_data_calculate`**

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

uint32_t	32-bit 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((uint32_t    *)    data_buffer,    BUFFER_SIZE,
INPUT_FORMAT_WORD);
```

## 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. Description of peripheral registers

DAC registers are listed in the table shown as below:

**Table 3-233. DAC Registers**

Register	Descriptions
DAC_CTL0	DACx control register 0
DAC_SWT	DACx software trigger register
DAC_OUT0_R12DH	DACx_OUT0 12-bit right-aligned data holding register
DAC_OUT0_L12DH	DACx_OUT0 12-bit left-aligned data holding register
DAC_OUT0_R8DH	DACx_OUT0 8-bit right-aligned data holding register
DAC_OUT1_R12DH	DACx_OUT1 12-bit right-aligned data holding register
DAC_OUT1_L12DH	DACx_OUT1 12-bit left-aligned data holding register
DAC_OUT1_R8DH	DACx_OUT1 8-bit right-aligned data holding register
DACC_R12DH	DACx concurrent mode 12-bit right-aligned data holding register
DACC_L12DH	DACx concurrent mode 12-bit left-aligned data holding register
DACC_R8DH	DACx concurrent mode 8-bit right-aligned data holding register
DAC_OUT0_DO	DACx_OUT0 data output register
DAC_OUT1_DO	DACx_OUT1 data output register
DAC_STAT0	DACx status register 0
DAC_CALR	DACx calibration register
DAC_MDCR	DACx mode control register
DAC_SKSTR0	DACx sample and keep sample time register 0

Register	Descriptions
DAC_SKSTR1	DACx sample and keep sample time register 1
DAC_SKKTR	DACx sample and keep time register
DAC_SKRTR	DACx sample and keep refresh time register
DAC_OUT0_SAW	DACx_OUT0 sawtooth register
DAC_OUT1_SAW	DACx_OUT1 sawtooth register
DAC_SAWMDR	DACx sawtooth mode register

### 3.8.2. Descriptions of Peripheral functions

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

**Table 3-234. DAC firmware functions**

Function name	Function description
dac_deinit	deinitialize DAC
dac_enable	enable DAC
dac_disable	disable DAC
dac_dma_enable	enable DAC DMA function
dac_dma_disable	disable DAC DMA function
dac_mode_config	configure DAC mode
dac_trimming_value_get	get the DACx trimming value
dac_trimming_value_set	set the DACx trimming value
dac_trimming_enable	enable the DACx trimming
dac_trimming_disable	disable the DACx trimming
dac_output_value_get	get DAC output value
dac_data_format_config	configure DAC data format
dac_data_set	set DAC data holding register value
dac_trigger_enable	enable DAC trigger
dac_trigger_disable	disable DAC trigger
dac_trigger_source_config	configure DAC trigger source
dac_software_trigger_enable	enable DAC software trigger
dac_wave_mode_config	configure DAC wave mode
dac_lfsr_noise_config	configure DAC LFSR noise mode
dac_triangle_noise_config	configure DAC triangle noise mode
dac_sawtooth_reset_trigger_source_config	configure DAC sawtooth reset trigger source
dac_sawtooth_step_trigger_source_config	configure DAC sawtooth step trigger source
dac_sawtooth_step_direction_config	configure DAC sawtooth step direction
dac_sawtooth_initial_value_set	set DAC sawtooth initial value
dac_sawtooth_step_value_set	set DAC sawtooth step value
dac_sawtooth_step_software_trigger_enable	enable DAC sawtooth step trigger

Function name	Function description
<code>dac_concurrent_data_set</code>	set DAC concurrent mode data holding register value
<code>dac_reset_persist_enable</code>	enable DAC reset persisting mode
<code>dac_reset_persist_disable</code>	disable DAC reset persisting mode
<code>dac_sample_keep_mode_config</code>	set DAC sample and keep time value
<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 3-235. Function `dac_deinit`**

<b>Function name</b>	<code>dac_deinit</code>
<b>Function prototype</b>	<code>void dac_deinit(uint32_t dac_periph);</code>
<b>Function descriptions</b>	deinitialize DAC
<b>Precondition</b>	-
<b>The called functions</b>	<code>rcu_periph_reset_enable</code> / <code>rcu_periph_reset_disable</code>
<b>Input parameter{in}</b>	
<b><code>dac_periph</code></b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### **dac\_enable**

The description of `dac_enable` is shown as below:

**Table 3-236. Function `dac_enable`**

<b>Function name</b>	<code>dac_enable</code>
<b>Function prototype</b>	<code>void dac_enable(uint32_t dac_periph, uint8_t dac_out);</code>
<b>Function descriptions</b>	enable DAC
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
Input parameter{in}	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### **dac\_disable**

The description of dac\_disable is shown as below:

**Table 3-237. Function dac\_disable**

<b>Function name</b>	dac_disable
<b>Function prototype</b>	void dac_disable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	disable DAC
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
Input parameter{in}	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC0_OUT0 */
dac_disable(DAC0, DAC_OUT0);
```

### **dac\_dma\_enable**

The description of dac\_dma\_enable is shown as below:

Table 3-238. Function `dac_dma_enable`

<b>Function name</b>	<code>dac_dma_enable</code>
<b>Function prototype</b>	<code>void dac_dma_enable(uint32_t dac_periph, uint8_t dac_out);</code>
<b>Function descriptions</b>	enable DAC DMA function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><code>dac_periph</code></b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b><code>dac_out</code></b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DAC0_OUT0 DMA function */
dac_dma_enable(DAC0, DAC_OUT0);
```

### **`dac_dma_disable`**

The description of `dac_dma_disable` is shown as below:

Table 3-239. Function `dac_dma_disable`

<b>Function name</b>	<code>dac_dma_disable</code>
<b>Function prototype</b>	<code>void dac_dma_disable(uint32_t dac_periph, uint8_t dac_out);</code>
<b>Function descriptions</b>	disable DAC DMA function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><code>dac_periph</code></b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b><code>dac_out</code></b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DAC0_OUT0 DMA function */
```

```
dac_dma_disable(DAC0, DAC_OUT0);
```

## dac\_mode\_config

The description of dac\_mode\_config is shown as below:

**Table 3-240. Function dac\_mode\_config**

<b>Function name</b>	dac_mode_config
<b>Function prototype</b>	void dac_mode_config(uint32_t dac_periph, uint32_t dac_out, uint32_t mode);
<b>Function descriptions</b>	configure DAC mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
DACx	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
DAC_OUTx	DAC output channel selection(x = 0,1)
<b>Input parameter{in}</b>	
<b>mode</b>	DAC working mode
NORMAL_PIN_BUFFON	DAC_OUTx work in normal mode and connect to external pin with buffer enable
NORMAL_PIN_PERIPH_BUFFON	DAC_OUTx work in normal mode and connect to external pin and on chip peripherals with buffer enable
NORMAL_PIN_BUFFOFF	DAC_OUTx work in normal mode and connect to external pin with buffer disable
NORMAL_PERIPH_BUFFOFF	DAC_OUTx work in normal mode and connect to on chip peripherals with buffer disable
SAMPLEKEEP_PIN_BUFFON	DAC_OUTx work in sample and keep mode and connect to external pin with buffer enable
SAMPLEKEEP_PIN_PERIPH_BUFFON	DAC_OUTx work in sample and keep mode and connect to external pin and on chip peripherals with buffer enable
SAMPLEKEEP_PIN_BUFFOFF	DAC_OUTx work in sample and keep mode and connect to external pin and on chip peripherals with buffer enable
SAMPLEKEEP_PERIPH_BUFFOFF	DAC_OUTx work in sample and keep mode and connect to on chip peripherals with buffer disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:



```
/* configure DAC0_OUT0 working in NORMAL_PIN_BUFFON mode */
```

```
dac_mode_config(DAC0, DAC_OUT0, NORMAL_PIN_BUFFON);
```

### **dac\_trimming\_value\_get**

The description of `dac_trimming_value_get` is shown as below:

**Table 3-241. Function `dac_trimming_value_get`**

<b>Function name</b>	<code>dac_trimming_value_get</code>
<b>Function prototype</b>	<code>void dac_trimming_value_set(uint32_t dac_periph, uint32_t dac_out,);</code>
<b>Function descriptions</b>	get the DACx trimming value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* get the DAC0_OUT0 trimming value */
```

```
data = dac_trimming_value_get (DAC0, DAC_OUT0);
```

### **dac\_trimming\_value\_set**

The description of `dac_trimming_value_set` is shown as below:

**Table 3-242. Function `dac_trimming_value_set`**

<b>Function name</b>	<code>dac_trimming_value_set</code>
<b>Function prototype</b>	<code>void dac_trimming_value_set(uint32_t dac_periph, uint32_t dac_out, uint32_t trim_value);</code>
<b>Function descriptions</b>	set the DACx trimming value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	

<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Input parameter{in}</b>	
<b>trim_value</b>	set new DAC trimming value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the DAC0_OUT0 trimming value */
dac_trimming_value_set(DAC0, DAC_OUT0, 1);
```

### **dac\_trimming\_enable**

The description of dac\_trimming\_enable is shown as below:

**Table 3-243. Function dac\_trimming\_enable**

<b>Function name</b>	dac_trimming_enable
<b>Function prototype</b>	void dac_trimming_enable(uint32_t dac_periph, uint32_t dac_out);
<b>Function descriptions</b>	enable the DACx trimming
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the DAC0_OUT0 trimming */
dac_trimming_enable (DAC0, DAC_OUT0);
```

### **dac\_trimming\_disable**

The description of dac\_trimming\_disable is shown as below:

Table 3-244. Function `dac_trimming_disable`

<b>Function name</b>	<code>dac_trimming_disable</code>
<b>Function prototype</b>	<code>void dac_trimming_disable(uint32_t dac_periph, uint32_t dac_out);</code>
<b>Function descriptions</b>	disable the DACx trimming
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><code>dac_periph</code></b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b><code>dac_out</code></b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the DAC0_OUT0 trimming */
dac_trimming_disable (DAC0, DAC_OUT0);
```

### **`dac_output_value_get`**

The description of `dac_output_value_get` is shown as below:

Table 3-245. Function `dac_output_value_get`

<b>Function name</b>	<code>dac_output_value_get</code>
<b>Function prototype</b>	<code>uint16_t dac_output_value_get(uint32_t dac_periph);</code>
<b>Function descriptions</b>	get DAC output value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><code>dac_periph</code></b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b><code>dac_out</code></b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b><code>uint16_t</code></b>	DAC output data (0~4095)

Example:

```
/* get the DAC0_OUT0 last data output value */
```

```
data = dac_output_value_get(DAC0, DAC_OUT0);
```

### **dac\_data\_format\_config**

The description of `dac_data_format_config` is shown as below:

**Table 3-246. Function `dac_data_format_config`**

Function name	<code>dac_data_format_config</code>
Function prototype	<code>void dac_data_format_config(uint32_t dac_periph, uint8_t dac_out, uint32_t format);</code>
Function descriptions	configure DAC data format
Precondition	-
The called functions	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
Input parameter{in}	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
Input parameter{in}	
<b>format</b>	DAC data format
<i>DAC_DATA_FORMAT_UNSIGNED</i>	unsigned data format
<i>DAC_DATA_FORMAT_SIGNED</i>	signed data format
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC0_OUT0 data signed format */
```

```
dac_data_format_config(DAC0, DAC_OUT0, DAC_DATA_FORMAT_UNSIGNED);
```

### **dac\_data\_set**

The description of `dac_data_set` is shown as below:

**Table 3-247. Function `dac_data_set`**

Function name	<code>dac_data_set</code>
Function prototype	<code>void dac_data_set(uint32_t dac_periph, uint8_t dac_out, uint32_t dac_align, uint16_t data);</code>
Function descriptions	set DAC data holding register value

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Input parameter{in}</b>	
<b>dac_align</b>	DAC data alignment mode
<i>DAC_ALIGN_8B_R</i>	8-bit right-aligned data
<i>DAC_ALIGN_12B_R</i>	12-bit right-aligned data
<i>DAC_ALIGN_12B_L</i>	12-bit left-aligned data
<b>Input parameter{in}</b>	
<b>data</b>	data to be loaded (0~4095)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set DAC0_OUT0 data holding register value */
```

```
dac_data_set(DAC0, DAC_OUT0, DAC_ALIGN_8B_R, 0xFF);
```

### **dac\_trigger\_enable**

The description of dac\_trigger\_enable is shown as below:

**Table 3-248. Function dac\_trigger\_enable**

<b>Function name</b>	dac_trigger_enable
<b>Function prototype</b>	void dac_trigger_enable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	enable DAC trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

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

### **dac\_trigger\_disable**

The description of dac\_trigger\_disable is shown as below:

**Table 3-249. Function dac\_trigger\_disable**

<b>Function name</b>	dac_trigger_disable
<b>Function prototype</b>	void dac_trigger_disable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	disable DAC trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DAC0_OUT0 trigger */
dac_trigger_disable(DAC0, DAC_OUT0);
```

### **dac\_trigger\_source\_config**

The description of dac\_trigger\_source\_config is shown as below:

**Table 3-250. Function dac\_trigger\_source\_config**

<b>Function name</b>	dac_trigger_source_config
<b>Function prototype</b>	void dac_trigger_source_config(uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);
<b>Function descriptions</b>	configure DAC trigger source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Input parameter{in}</b>	
<b>triggersource</b>	external trigger of DAC
<i>DAC_TRIGGER_T5_TRG</i> 0	TIMER5 TRGO
<i>DAC_TRIGGER_T14_TRG</i> 0	TIMER14 TRGO
<i>DAC_TRIGGER_T6_TRG</i> 0	TIMER6 TRGO
<i>DAC_TRIGGER_T2_TRG</i> 0	TIMER2 TRGO
<i>DAC_TRIGGER_T1_TRG</i> 0	TIMER1 TRGO
<i>DAC_TRIGGER_T0_TRG</i> 0	TIMER0 TRGO
<i>DAC_TRIGGER_EXTI_9</i>	EXTI interrupt line9 event
<i>DAC_TRIGGER_SOFTWARE</i>	software trigger
<i>DAC_TRIGGER_CLA0</i>	CLA0 trigger
<i>DAC_TRIGGER_CLA1</i>	CLA1 trigger
<i>DAC_TRIGGER_CLA2</i>	CLA2 trigger
<i>DAC_TRIGGER_CLA3</i>	CLA3 trigger
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DAC0_OUT0 trigger source */
```

```
dac_trigger_source_config(DAC0, DAC_OUT0, DAC_TRIGGER_T1_TRGO);
```

### **dac\_software\_trigger\_enable**

The description of `dac_trigger_source_enable` is shown as below:

**Table 3-251. Function `dac_software_trigger_enable`**

<b>Function name</b>	<code>dac_software_trigger_enable</code>
<b>Function prototype</b>	<code>void dac_software_trigger_enable(uint32_t dac_periph, uint8_t dac_out);</code>

<b>Function descriptions</b>	enable DAC software trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
dac_software_trigger_enable(DAC0, DAC_OUT0);
```

### **dac\_wave\_mode\_config**

The description of dac\_wave\_mode\_config is shown as below:

**Table 3-252. Function dac\_wave\_mode\_config**

<b>Function name</b>	dac_wave_mode_config
<b>Function prototype</b>	void dac_wave_mode_config(uint32_t dac_periph, uint8_t dac_out, uint32_t wave_mode);
<b>Function descriptions</b>	configure DAC wave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Input parameter{in}</b>	
<b>wave_mode</b>	DAC wave mode
<i>DAC_WAVE_DISABLE</i>	wave mode disable
<i>DAC_WAVE_MODE_LFSR</i>	LFSR noise mode
<i>DAC_WAVE_MODE_TRIANGLE</i>	triangle noise mode
<b>Output parameter{out}</b>	
-	-



Return value	
-	-

Example:

```
/* configure DAC0_OUT0 wave mode */
```

```
dac_wave_mode_config(DAC0, DAC_OUT0, DAC_WAVE_DISABLE);
```

### **dac\_lfsr\_noise\_config**

The description of `dac_lfsr_noise_config` is shown as below:

**Table 3-253. Function `dac_lfsr_noise_config`**

Function name	<code>dac_lfsr_noise_config</code>
Function prototype	<code>void dac_lfsr_noise_config(uint32_t dac_periph, uint8_t dac_out, uint32_t unmask_bits);</code>
Function descriptions	configure DAC LFSR noise mode
Precondition	-
The called functions	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
Input parameter{in}	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
Input parameter{in}	
<b>unmask_bits</b>	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 = 1,2,3..11)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DAC0_OUT0 LFSR noise mode */
```

```
dac_lfsr_noise_config(DAC0, DAC_OUT0, DAC_LFSR_BIT0);
```

### **dac\_triangle\_noise\_config**

The description of `dac_triangle_noise_config` is shown as below:

**Table 3-254. Function `dac_triangle_noise_config`**

Function name	<code>dac_triangle_noise_config</code>
---------------	--

<b>Function prototype</b>	void dac_triangle_noise_config(uint32_t dac_periph, uint8_t dac_out, uint32_t amplitude);
<b>Function descriptions</b>	configure DAC triangle noise mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Input parameter{in}</b>	
<b>amplitude</b>	the amplitude of the triangle
<i>DAC_TRIANGLE_AMPLITUDE_x</i>	$x = 2^n - 1$ (n = 1..12)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DAC0_OUT0 triangle noise mode */
```

```
dac_triangle_noise_config(DAC0, DAC_OUT0, DAC_TRIANGLE_AMPLITUDE_1);
```

### **dac\_sawtooth\_reset\_trigger\_source\_config**

The description of dac\_sawtooth\_reset\_trigger\_source\_config is shown as below:

**Table 3-255. Function dac\_sawtooth\_reset\_trigger\_source\_config**

<b>Function name</b>	dac_sawtooth_reset_trigger_source_config
<b>Function prototype</b>	void dac_sawtooth_reset_trigger_source_config(uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);
<b>Function descriptions</b>	configure DAC sawtooth reset trigger source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Input parameter{in}</b>	
<b>triggersource</b>	sawtooth reset trigger source of DAC

<i>DAC_TRIGGER_EXTERNAL</i>	external trigger from TRIGSEL
<i>DAC_TRIGGER_SOFTWARE</i>	software trigger
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DAC0_OUT0 sawtooth reset trigger source */
```

```
dac_sawtooth_reset_trigger_source_config(DAC0,DAC_OUT0,DAC_TRIGGER_EXTERNAL);
```

### **dac\_sawtooth\_step\_trigger\_source\_config**

The description of `dac_sawtooth_step_trigger_source_config` is shown as below:

**Table 3-256. Function `dac_sawtooth_step_trigger_source_config`**

<b>Function name</b>	<code>dac_sawtooth_step_trigger_source_config</code>
<b>Function prototype</b>	<code>void dac_sawtooth_step_trigger_source_config(uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);</code>
<b>Function descriptions</b>	configure DAC sawtooth step trigger source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Input parameter{in}</b>	
<b>triggersource</b>	sawtooth step trigger source of DAC
<i>DAC_TRIGGER_EXTERNAL</i>	external trigger from TRIGSEL
<i>DAC_TRIGGER_SOFTWARE</i>	software trigger
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DAC0_OUT0 sawtooth step trigger source */
```

```
dac_sawtooth_step_trigger_source_config(DAC0,DAC_OUT0,DAC_TRIGGER_EXTERNAL);
```

### **dac\_sawtooth\_step\_direction\_config**

The description of `dac_sawtooth_step_direction_config` is shown as below:

**Table 3-257. Function `dac_sawtooth_step_direction_config`**

<b>Function name</b>	<code>dac_sawtooth_step_direction_config</code>
<b>Function prototype</b>	<code>void dac_sawtooth_step_direction_config (uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);</code>
<b>Function descriptions</b>	configure DAC sawtooth step direction
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Input parameter{in}</b>	
<b>direction</b>	sawtooth step direction of DAC
<i>DAC_SAWTOOTH_STEP_DOWN</i>	sawtooth step down
<i>DAC_SAWTOOTH_STEP_UP</i>	sawtooth step up
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DAC0_OUT0 sawtooth step direction */
```

```
dac_sawtooth_step_direction_config(DAC0,DAC_OUT0,DAC_SAWTOOTH_STEP_DOWN);
```

### **dac\_sawtooth\_initial\_value\_set**

The description of `dac_sawtooth_initial_value_set` is shown as below:

**Table 3-258. Function `dac_sawtooth_initial_value_set`**

<b>Function name</b>	<code>dac_sawtooth_initial_value_set</code>
<b>Function prototype</b>	<code>void dac_sawtooth_initial_value_set(uint32_t dac_periph, uint8_t dac_out, uint32_t init_value);</code>

<b>Function descriptions</b>	set the DACx sawtooth initial value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Input parameter{in}</b>	
<b>init_value</b>	set DAC sawtooth initial value(0~4095)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set DAC0_OUT0 sawtooth initial value */
```

```
dac_sawtooth_initial_value_set(DAC0, DAC_OUT0, 0x0F);
```

### **dac\_sawtooth\_step\_value\_set**

The description of dac\_sawtooth\_step\_value\_set is shown as below:

**Table 3-259. Function dac\_sawtooth\_step\_value\_set**

<b>Function name</b>	dac_sawtooth_step_value_set
<b>Function prototype</b>	void dac_sawtooth_step_value_set(uint32_t dac_periph, uint8_t dac_out, uint32_t step_value);
<b>Function descriptions</b>	set the DACx sawtooth step value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Input parameter{in}</b>	
<b>step_value</b>	set DAC sawtooth step value(0x0~0xFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set DAC0_OUT0 sawtooth step value */
dac_sawtooth_step_value_set(DAC0, DAC_OUT0, 0x0F);
```

### **dac\_sawtooth\_step\_software\_trigger\_enable**

The description of dac\_sawtooth\_step\_software\_trigger\_enable is shown as below:

**Table 3-260. Function dac\_sawtooth\_step\_software\_trigger\_enable**

<b>Function name</b>	dac_sawtooth_step_software_trigger_enable
<b>Function prototype</b>	void dac_sawtooth_step_software_trigger_enable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	enable DAC sawtooth step software trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DAC0_OUT0 sawtooth step trigger */
dac_sawtooth_step_software_trigger_enable(DAC0, DAC_OUT0);
```

### **dac\_concurrent\_enable**

The description of dac\_concurrent\_enable is shown as below:

**Table 3-261. Function dac\_concurrent\_enable**

<b>Function name</b>	dac_concurrent_enable
<b>Function prototype</b>	void dac_concurrent_enable(uint32_t dac_periph);
<b>Function descriptions</b>	enable DAC concurrent mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC0 concurrent mode */
dac_concurrent_enable(DAC0);
```

### **dac\_concurrent\_disable**

The description of dac\_concurrent\_disable is shown as below:

**Table 3-262. Function dac\_concurrent\_disable**

Function name	dac_concurrent_disable
Function prototype	void dac_concurrent_disable(uint32_t dac_periph);
Function descriptions	disable DAC concurrent mode
Precondition	-
The called functions	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DAC0 concurrent mode */
dac_concurrent_disable(DAC0);
```

### **dac\_concurrent\_software\_trigger\_enable**

The description of dac\_concurrent\_software\_trigger\_enable is shown as below:

**Table 3-263. Function dac\_concurrent\_software\_trigger\_enable**

Function name	dac_concurrent_software_trigger_enable
Function prototype	void dac_concurrent_software_trigger_enable(uint32_t dac_periph);
Function descriptions	enable DAC concurrent software trigger
Precondition	-
The called functions	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral

<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DAC0 concurrent software trigger */
```

```
dac_concurrent_software_trigger_enable(DAC0);
```

### **dac\_concurrent\_data\_set**

The description of `dac_concurrent_data_set` is shown as below:

**Table 3-264. Function `dac_concurrent_data_set`**

<b>Function name</b>	<code>dac_concurrent_data_set</code>
<b>Function prototype</b>	<code>void dac_concurrent_data_set(uint32_t dac_periph, uint32_t dac_align, uint16_t data0, uint16_t data1);</code>
<b>Function descriptions</b>	set DAC concurrent mode data holding register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_align</b>	DAC data alignment mode
<i>DAC_ALIGN_8B_R</i>	8-bit right-aligned data
<i>DAC_ALIGN_12B_R</i>	12-bit right-aligned data
<i>DAC_ALIGN_12B_L</i>	12-bit left-aligned data
<b>Input parameter{in}</b>	
<b>data0</b>	DAC0 data to be loaded (0~4095)
<b>Input parameter{in}</b>	
<b>data1</b>	DAC1 data to be loaded (0~4095)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set DAC0 concurrent mode data holding register value */
```

```
dac_concurrent_data_set(DAC0, DAC_ALIGN_8B_R, 0xFF, 0xFF);
```



## dac\_reset\_persist\_enable

The description of dac\_reset\_persist\_enable is shown as below:

**Table 3-265. Function dac\_reset\_persist\_enable**

<b>Function name</b>	dac_reset_persist_enable
<b>Function prototype</b>	void dac_reset_persist_enable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	enable DAC reset persisting mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DAC0_OUT0 reset persisting mode */
dac_reset_persist_enable(DAC0, DAC_OUT0);
```

## dac\_reset\_persist\_disable

The description of dac\_reset\_persist\_disable is shown as below:

**Table 3-266. Function dac\_reset\_persist\_disable**

<b>Function name</b>	dac_reset_persist_disable
<b>Function prototype</b>	void dac_reset_persist_disable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	disable DAC reset persisting mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection (x = 0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable DAC0_OUT0 reset persisting mode */
dac_reset_persist_disable(DAC0, DAC_OUT0);
```

### **dac\_sample\_keep\_mode\_config**

The description of dac\_sample\_keep\_mode\_config is shown as below:

**Table 3-267. Function dac\_sample\_keep\_mode\_config**

Function name	dac_sample_keep_mode_config
Function prototype	void dac_sample_keep_mode_config(uint32_t dac_periph, uint32_t dac_out, uint32_t sample_time, uint32_t keep_time, uint32_t refresh_time);
Function descriptions	set DAC sample and keep time value
Precondition	-
The called functions	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
Input parameter{in}	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
Input parameter{in}	
<b>sample_time</b>	DAC sample time(0~1023)
Input parameter{in}	
<b>keep_time</b>	DAC keep time(0~1023)
Input parameter{in}	
<b>refresh_time</b>	DAC refresh time(0~255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC0_OUT0 sample and keep time value */
dac_sample_keep_mode_config (DAC0, DAC_OUT0, 1, 1, 1);
```

### **dac\_flag\_get**

The description of dac\_flag\_get is shown as below:

Table 3-268. Function `dac_flag_get`

<b>Function name</b>	<code>dac_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus dac_flag_get(uint32_t dac_periph, uint32_t flag);</code>
<b>Function descriptions</b>	get DAC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><code>dac_periph</code></b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b><code>flag</code></b>	the DAC status flags
<i>DAC_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun flag
<i>DAC_FLAG_CALF0</i>	DACx_OUT0 calibration offset flag
<i>DAC_FLAG_BWT0</i>	DACx_OUT0 sample and keep write enable flag
<i>DAC_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun flag
<i>DAC_FLAG_CALF1</i>	DACx_OUT1 calibration offset flag
<i>DAC_FLAG_BWT1</i>	DACx_OUT1 sample and keep write enable flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	the state of DAC bit (SET or RESET)

Example:

```
/* get DAC0 flag */
```

```
flag = dac_flag_get(DAC0, DAC_FLAG_DDUDR0);
```

### **`dac_flag_clear`**

The description of `dac_flag_clear` is shown as below:

Table 3-269. Function `dac_flag_clear`

<b>Function name</b>	<code>dac_flag_clear</code>
<b>Function prototype</b>	<code>void dac_flag_clear(uint32_t dac_periph, uint32_t flag);</code>
<b>Function descriptions</b>	clear DAC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><code>dac_periph</code></b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b><code>flag</code></b>	DAC flag
<i>DAC_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun flag
<i>DAC_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun flag

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DAC0 flag */
dac_flag_clear(DAC0, DAC_FLAG_DDUDR0);
```

### **dac\_interrupt\_enable**

The description of dac\_interrupt\_enable is shown as below:

**Table 3-270. Function dac\_interrupt\_enable**

Function name	dac_interrupt_enable
Function prototype	void dac_interrupt_enable(uint32_t dac_periph, uint32_t interrupt);
Function descriptions	enable DAC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dac_periph	DAC peripheral
DACx	DAC peripheral selection(x = 0,1,2,3)
Input parameter{in}	
interrupt	the DAC interrupt
DAC_INT_DDUDRIE0	DACx_OUT0 DMA underrun interrupt
DAC_INT_DDUDRIE1	DACx_OUT1 DMA underrun interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DAC0 interrupt */
dac_interrupt_enable (DAC0, DAC_INT_DDUDRIE0);
```

### **dac\_interrupt\_disable**

The description of dac\_interrupt\_disable is shown as below:

**Table 3-271. Function dac\_interrupt\_disable**

Function name	dac_interrupt_disable
Function prototype	void dac_interrupt_disable(uint32_t dac_periph, uint32_t interrupt);
Function descriptions	disable DAC interrupt

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>interrupt</b>	the DAC interrupt
<i>DAC_INT_DDUDRIE0</i>	DACx_OUT0 DMA underrun interrupt
<i>DAC_INT_DDUDRIE1</i>	DACx_OUT1 DMA underrun interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DAC0 interrupt */
```

```
dac_interrupt_disable (DAC0, DAC_INT_DDUDRIE0);
```

### **dac\_interrupt\_flag\_get**

The description of dac\_interrupt\_flag\_get is shown as below:

**Table 3-272. Function dac\_interrupt\_flag\_get**

<b>Function name</b>	dac_interrupt_flag_get
<b>Function prototype</b>	FlagStatus dac_interrupt_flag_get(uint32_t dac_periph, uint32_t int_flag);
<b>Function descriptions</b>	get DAC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
<b>Input parameter{in}</b>	
<b>int_flag</b>	DAC interrupt flag
<i>DAC_INT_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt flag
<i>DAC_INT_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	the state of DAC interrupt flag(SET or RESET)

Example:

```
/* get DAC0 interrupt flag */
```

```
flag = dac_interrupt_flag_get(DAC0, DAC_INT_FLAG_DDUDR0);
```

## **dac\_interrupt\_flag\_clear**

The description of `dac_interrupt_flag_clear` is shown as below:

**Table 3-273. Function `dac_interrupt_flag_clear`**

Function name	<code>dac_interrupt_flag_clear</code>
Function prototype	<code>FlagStatus dac_interrupt_flag_clear(uint32_t dac_periph, uint32_t int_flag);</code>
Function descriptions	clear DAC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0,1,2,3)
Input parameter{in}	
<b>int_flag</b>	DAC interrupt flag
<i>DAC_INT_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt flag
<i>DAC_INT_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DAC0 interrupt flag */
```

```
dac_interrupt_flag_clear(DAC0, DAC_INT_FLAG_DDUDR0);
```

## **3.9. DBG**

The DBG hold unit helps debugger to debug power saving mode. The DBG registers are listed in chapter [3.9.1](#). the DBG firmware functions are introduced in chapter [3.9.2](#).

### **3.9.1. Descriptions of Peripheral registers**

DBG registers are listed in the table shown as below:

**Table 3-274. DBG Registers**

Registers	Descriptions
DBG_ID	DBG ID code register
DBG_CTL0	DBG control register 0
DBG_CTL1	DBG control register 1

Registers	Descriptions
DBG_CTL2	DBG control register 2

### 3.9.2. Descriptions of Peripheral functions

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

**Table 3-275. 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_trace_pin_enable	enable trace pin assignment
dbg_trace_pin_disable	disable trace pin assignment
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 3-276. 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_TIMERx_HOLD	hold TIMERx (x=0,1,2,3,4,5,6,7,14,15,16,19) counter when core is halted
DBG_I2Cx_HOLD	hold I2Cx (x=0,1,2,3) smbus when core is halted
DBG_CANx_HOLD	hold CANx (x=0,1,2) counter when core is halted
DBG_RTC_HOLD	hold RTC calendar and wakeup counter when core is halted
DBG_LPTIMER_HOLD	debug LPTIMER kept when core is halted
DBG_HRTIMER_HOLD	debug HRTIMER kept when core is halted

### dbg\_deinit

The description of dbg\_deinit is shown as below:

**Table 3-277. 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 3-278. 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 3-279. 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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-280. Function dbg\_low\_power\_disable**

<b>Function name</b>	dbg_low_power_disable
<b>Function prototype</b>	void dbg_low_power_disable(uint32_t dbg_low_power);
<b>Function descriptions</b>	disable low power behavior when the mcu is in debug mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dbg_low_power</b>	low power mode
<i>DBG_LOW_POWER_SLEEP</i>	do not keep debugger connection during sleep mode
<i>DBG_LOW_POWER_DEEPSLEEP</i>	do not keep debugger connection during deepsleep mode
<i>DBG_LOW_POWER_STANDBY</i>	do not keep debugger connection during standby mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
dbg_low_power_disable(DBG_LOW_POWER_SLEEP);
```

## dbg\_trace\_pin\_enable

The description of dbg\_trace\_pin\_enable is shown as below:

**Table 3-281. Function dbg\_trace\_pin\_enable**

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

Example:

```
/* enable trace pin assignment */
dbg_trace_pin_enable();
```

## dbg\_trace\_pin\_disable

The description of dbg\_trace\_pin\_disable is shown as below:

**Table 3-282. Function dbg\_trace\_pin\_disable**

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

Example:

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

## dbg\_periph\_enable

The description of dbg\_periph\_enable is shown as below:

**Table 3-283. Function dbg\_periph\_enable**

<b>Function name</b>	dbg_periph_enable
<b>Function prototype</b>	void dbg_periph_enable(dbg_periph_enum dbg_periph);
<b>Function descriptions</b>	Enable peripheral behavior when the mcu is in debug mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dbg_periph</b>	Peripheral refer to <a href="#">Table 3-276. Enum dbg_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-284. Function dbg\_periph\_disable**

<b>Function name</b>	dbg_periph_disable
<b>Function prototype</b>	void dbg_periph_disable(dbg_periph_enum dbg_periph);
<b>Function descriptions</b>	Disable peripheral behavior when the mcu is in debug mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dbg_periph</b>	peripheral refer to <a href="#">Table 3-276. Enum dbg_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
dbg_periph_disable(DBG_TIMER1_HOLD);
```

## 3.10. 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 [3.10.1](#), the DMA firmware functions are introduced in chapter [3.10.2](#)

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 [3.10.1](#), the DMAMUX firmware functions are introduced in chapter [3.10.2](#)

### 3.10.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

**Table 3-285. DMA Registers**

Registers	Descriptions
DMA_INTF	Interrupt flag register
DMA_INTC	Interrupt flag clear register
DMA_CHxCTL (x=0..6)	Channel x control register
DMA_CHxCNT (x=0..6)	Channel x counter register
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 3-286. DMAMUX Registers**

Registers	Descriptions
DMAMUX_RM_CHx CFG (x=0..13)	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	Request generator channel interrupt flag clear register

Registers	Descriptions
C	

### 3.10.2. Descriptions of Peripheral functions

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

**Table 3-287. DMA firmware function**

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

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

**Table 3-288. DMAMUX firmware function**

Function name	Function description
<code>dmamux_sync_struct_para_init</code>	initialize the parameters of DMAMUX synchronization mode

Function name	Function description
	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
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 3-289. 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 3-290. 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 3-291. 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 3-292. 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 3-293. 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
DMAMUX_MUXCH 12	DMAMUX request multiplexer Channel 12
DMAMUX_MUXCH 13	DMAMUX request multiplexer Channel 13

### Enum dmamux\_generator\_channel\_enum

**Table 3-294. 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 3-295. Enum dmamux\_interrupt\_enum**

Member name	Function description
DMAMUX_INT_MUXCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
DMAMUX_INT_MUXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
DMAMUX_INT_MUXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun interrupt
DMAMUX_INT_MUXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
DMAMUX_INT_MUXCH4_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_MU XCH12_SO	DMAMUX request multiplexer channel 12 synchronization overrun interrupt
DMAMUX_INT_MU XCH13_SO	DMAMUX request multiplexer channel 13 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 3-296. Enum dmamux\_flag\_enum**

Member name	Function description
DMAMUX_FLAG_M UXCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun flag
DMAMUX_FLAG_M UXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun flag
DMAMUX_FLAG_M UXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun flag
DMAMUX_FLAG_M UXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun flag
DMAMUX_FLAG_M UXCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun flag
DMAMUX_FLAG_M UXCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun flag

DMAMUX_FLAG_M UXCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun flag
DMAMUX_FLAG_M UXCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun flag
DMAMUX_FLAG_M UXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun flag
DMAMUX_FLAG_M UXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun flag
DMAMUX_FLAG_M UXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun flag
DMAMUX_FLAG_M UXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun flag
DMAMUX_FLAG_M UXCH12_SO	DMAMUX request multiplexer channel 12 synchronization overrun flag
DMAMUX_FLAG_M UXCH13_SO	DMAMUX request multiplexer channel 13 synchronization overrun flag
DMAMUX_FLAG_G ENCH0_TO	DMAMUX request generator channel 0 trigger overrun flag
DMAMUX_FLAG_G ENCH1_TO	DMAMUX request generator channel 1 trigger overrun flag
DMAMUX_FLAG_G ENCH2_TO	DMAMUX request generator channel 2 trigger overrun flag
DMAMUX_FLAG_G ENCH3_TO	DMAMUX request generator channel 3 trigger overrun flag

### Enum dmamux\_interrupt\_flag\_enum

**Table 3-297. 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_MUXCH12_SO	DMAMUX request multiplexer channel 12 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH13_SO	DMAMUX request multiplexer channel 13 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 3-298. Function dma\_deinit**

<b>Function name</b>	dma_deinit
<b>Function prototype</b>	void dma_deinit(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	deinitialize DMA a channel registers
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<i>DMA_CHx</i>	DMA channel selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### **dma\_struct\_para\_init**

The description of dma\_struct\_para\_init is shown as below:

**Table 3-299. Function dma\_struct\_para\_init**

<b>Function name</b>	dma_struct_para_init
<b>Function prototype</b>	void dma_struct_para_init(dma_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of DMA struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>*init_struct</b>	A dma_parameter_struct address, refer to <a href="#">Table 3-289. Structure dma_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-300. Function dma\_init**

<b>Function name</b>	dma_init
<b>Function prototype</b>	void dma_init(uint32_t dma_periph, dma_channel_enum channelx, dma_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize DMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<i>DMA_CHx</i>	DMA channel selection

Input parameter{in}	
<b>init_struct</b>	A dma_parameter_struct address, the structure members refer to <a href="#">Table 3-289. Structure dma_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

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

dma_init_struct.direction = DMA_PERIPHERAL_TO_MEMORY;
dma_init_struct.memory_addr = (uint32_t)g_destbuf;
dma_init_struct.memory_inc = DMA_MEMORY_INCREASE_ENABLE;
dma_init_struct.memory_width = DMA_MEMORY_WIDTH_32BIT;
dma_init_struct.number = TRANSFER_NUM/4;
dma_init_struct.periph_addr = (uint32_t)FLASH_WRITE_START_ADDR;
dma_init_struct.periph_inc = DMA_PERIPH_INCREASE_ENABLE;
dma_init_struct.periph_width = DMA_PERIPHERAL_WIDTH_32BIT;
dma_init_struct.priority = DMA_PRIORITY_ULTRA_HIGH;
dma_init(DMA0, DMA_CH0, &dma_init_struct);

```

### **dma\_circulation\_enable**

The description of dma\_circulation\_enable is shown as below:

**Table 3-301. Function dma\_circulation\_enable**

<b>Function name</b>	dma_circulation_enable
<b>Function prototype</b>	void dma_circulation_enable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	enable DMA circulation mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Output parameter{out}	
-	-

Return value	
-	-

Example:

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

### **dma\_circulation\_disable**

The description of dma\_circulation\_disable is shown as below:

**Table 3-302. Function dma\_circulation\_disable**

<b>Function name</b>	dma_circulation_disable
<b>Function prototype</b>	void dma_circulation_disable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	disable DMA circulation mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### **dma\_memory\_to\_memory\_enable**

The description of dma\_memory\_to\_memory\_enable is shown as below:

**Table 3-303. Function dma\_memory\_to\_memory\_enable**

<b>Function name</b>	dma_memory_to_memory_enable
<b>Function prototype</b>	void dma_memory_to_memory_enable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	enable memory to memory mode
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### **dma\_memory\_to\_memory\_disable**

The description of dma\_memory\_to\_memory\_disable is shown as below:

**Table 3-304. Function dma\_memory\_to\_memory\_disable**

<b>Function name</b>	dma_memory_to_memory_disable
<b>Function prototype</b>	void dma_memory_to_memory_disable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	disable memory to memory mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### **dma\_channel\_enable**

The description of dma\_channel\_enable is shown as below:

Table 3-305. Function dma\_channel\_enable

Function name	dma_channel_enable
Function prototype	void dma_channel_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### dma\_channel\_disable

The description of dma\_channel\_disable is shown as below:

Table 3-306. Function dma\_channel\_disable

Function name	dma_channel_disable
Function prototype	void dma_channel_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-



Example:

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

### **dma\_periph\_address\_config**

The description of dma\_periph\_address\_config is shown as below:

**Table 3-307. Function dma\_periph\_address\_config**

<b>Function name</b>	dma_periph_address_config
<b>Function prototype</b>	void dma_periph_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t address);
<b>Function descriptions</b>	set DMA peripheral base address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>address</b>	peripheral base address
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### **dma\_memory\_address\_config**

The description of dma\_memory\_address\_config is shown as below:

**Table 3-308. Function dma\_memory\_address\_config**

<b>Function name</b>	dma_memory_address_config
<b>Function prototype</b>	void dma_memory_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t address);
<b>Function descriptions</b>	set DMA memory base address
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Input parameter{in}	
<b>address</b>	memory base address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
uint8_t g_destbuf[TRANSFER_NUM];
```

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

### **dma\_transfer\_number\_config**

The description of dma\_transfer\_number\_config is shown as below:

**Table 3-309. Function dma\_transfer\_number\_config**

<b>Function name</b>	dma_transfer_number_config
<b>Function prototype</b>	void dma_transfer_number_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t number);
<b>Function descriptions</b>	set the number of remaining data to be transferred by the DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Input parameter{in}	
<b>number</b>	data transfer number (0x00000000 – 0x0000FFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
#define TRANSFER_NUM                0x400
```

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

## dma\_transfer\_number\_get

The description of dma\_transfer\_number\_get is shown as below:

**Table 3-310. Function dma\_transfer\_number\_get**

<b>Function name</b>	dma_transfer_number_get
<b>Function prototype</b>	uint32_t dma_transfer_number_get(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	get the number of remaining data to be transferred by the DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	0x00000000 – 0x0000FFFF

Example:

```
uint32_t number = 0;
```

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

## dma\_priority\_config

The description of dma\_priority\_config is shown as below:

**Table 3-311. Function dma\_priority\_config**

<b>Function name</b>	dma_priority_config
<b>Function prototype</b>	void dma_priority_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t priority);
<b>Function descriptions</b>	configure priority level of DMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel

<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>priority</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### dma\_memory\_width\_config

The description of dma\_memory\_width\_config is shown as below:

**Table 3-312. Function dma\_memory\_width\_config**

<b>Function name</b>	dma_memory_width_config
<b>Function prototype</b>	void dma_memory_width_config (uint32_t dma_periph, dma_channel_enum channelx, uint32_t mwidth);
<b>Function descriptions</b>	configure transfer data size of memory
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>mwidth</b>	transfer data width of memory
<i>DMA_MEMORY_WIDTH_8BIT</i>	transfer data width of memory is 8-bit
<i>DMA_MEMORY_WIDTH_16BIT</i>	transfer data width of memory is 16-bit
<i>DMA_MEMORY_WIDTH_32BIT</i>	transfer data width of memory is 32-bit
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

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

### dma\_periph\_width\_config

The description of dma\_periph\_width\_config is shown as below:

**Table 3-313. Function dma\_periph\_width\_config**

<b>Function name</b>	dma_periph_width_config
<b>Function prototype</b>	void dma_periph_width_config (uint32_t dma_periph, dma_channel_enum channelx, uint32_t pwidth);
<b>Function descriptions</b>	configure transfer data size of peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>pwidth</b>	transfer data width of peripheral
<i>DMA_PERIPHERAL_WIDTH_8BIT</i>	transfer data width of peripheral is 8-bit
<i>DMA_PERIPHERAL_WIDTH_16BIT</i>	transfer data width of peripheral is 16-bit
<i>DMA_PERIPHERAL_WIDTH_32BIT</i>	transfer data width of peripheral is 32-bit
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### dma\_memory\_increase\_enable

The description of dma\_memory\_increase\_enable is shown as below:

Table 3-314. Function dma\_memory\_increase\_enable

Function name	dma_memory_increase_enable
Function prototype	void dma_memory_increase_enable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	enable next address increasement algorithm of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### dma\_memory\_increase\_disable

The description of dma\_memory\_increase\_disable is shown as below:

Table 3-315. Function dma\_memory\_increase\_disable

Function name	dma_memory_increase_disable
Function prototype	void dma_memory_increase_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable next address increasement algorithm of memory
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### dma\_periph\_increase\_enable

The description of dma\_periph\_increase\_enable is shown as below:

**Table 3-316. Function dma\_periph\_increase\_enable**

<b>Function name</b>	dma_periph_increase_enable
<b>Function prototype</b>	void dma_periph_increase_enable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	enable next address increasement algorithm of peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### dma\_periph\_increase\_disable

The description of dma\_periph\_increase\_disable is shown as below:

**Table 3-317. Function dma\_periph\_increase\_disable**

<b>Function name</b>	dma_periph_increase_disable
<b>Function prototype</b>	void dma_periph_increase_disable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	disable next address increasement algorithm of peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
dma_periph_increase_disable(DMA0, DMA_CH0);
```

### **dma\_transfer\_direction\_config**

The description of dma\_transfer\_direction\_config is shown as below:

**Table 3-318. Function dma\_transfer\_direction\_config**

Function name	dma_transfer_direction_config
Function prototype	void dma_transfer_direction_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t direction);
Function descriptions	configure the direction of data transfer on the channel
Precondition	-
The called functions	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Input parameter{in}	
<b>direction</b>	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:

```
dma_transfer_direction_config(DMA0, DMA_CH0, DMA_PERIPHERAL_TO_MEMORY);
```

### **dma\_flag\_get**

The description of dma\_flag\_get is shown as below:



Table 3-319. Function dma\_flag\_get

Function name	dma_flag_get
Function prototype	FlagStatus dma_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	check DMA flag is set or not
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
Input parameter{in}	
flag	specify get which flag
DMA_FLAG_G	global interrupt flag of channel
DMA_FLAG_FTF	full transfer finish flag of channel
DMA_FLAG_HTF	half transfer finish flag of channel
DMA_FLAG_ERR	error flag of channel
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus flag = RESET;
```

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

### dma\_flag\_clear

The description of dma\_flag\_clear is shown as below:

Table 3-320. Function dma\_flag\_clear

Function name	dma_flag_clear
Function prototype	void dma_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
Function descriptions	clear the flag of a DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	DMA peripheral selection
Input parameter{in}	

<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### dma\_interrupt\_enable

The description of dma\_interrupt\_enable is shown as below:

**Table 3-321. Function dma\_interrupt\_enable**

<b>Function name</b>	dma_interrupt_enable
<b>Function prototype</b>	void dma_interrupt_enable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t source);
<b>Function descriptions</b>	enable DMA interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>source</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* DMA0 channel0 interrupt configuration */
```

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

### dma\_interrupt\_disable

The description of dma\_interrupt\_disable is shown as below:

**Table 3-322. Function dma\_interrupt\_disable**

<b>Function name</b>	dma_interrupt_disable
<b>Function prototype</b>	void dma_interrupt_disable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t source);
<b>Function descriptions</b>	disable DMA interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>source</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* DMA0 channel0 interrupt configuration */
```

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

### dma\_interrupt\_flag\_get

The description of dma\_interrupt\_flag\_get is shown as below:

**Table 3-323. Function dma\_interrupt\_flag\_get**

<b>Function name</b>	dma_interrupt_flag_get
<b>Function prototype</b>	FlagStatus dma_interrupt_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
<b>Function descriptions</b>	check DMA flag and interrupt enable bit is set or not

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

## dma\_interrupt\_flag\_clear

The description of dma\_interrupt\_flag\_clear is shown as below:

**Table 3-324. Function dma\_interrupt\_flag\_clear**

<b>Function name</b>	dma_interrupt_flag_clear
<b>Function prototype</b>	void dma_interrupt_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
<b>Function descriptions</b>	clear the interrupt flag of a DMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx</i>	DMA channel selection, refer to <a href="#">Table 3-292. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### dmamux\_sync\_struct\_para\_init

The description of dmamux\_sync\_struct\_para\_init is shown as below:

**Table 3-325. Function dmamux\_sync\_struct\_para\_init**

<b>Function name</b>	dmamux_sync_struct_para_init
<b>Function prototype</b>	void dmamux_sync_struct_para_init(dmamux_sync_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize the parameters of DMAMUX synchronization mode structure with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>init_struct</b>	the initialization data needed to initialize DMAMUX request multiplexer channel synchronization mode, refer to <a href="#">Table 3-290. Structure dmamux_sync_parameter_struct</a>
<b>Return value</b>	
-	-

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 3-326. Function `dmamux_synchronization_init`

Function name	<code>dmamux_synchronization_init</code>
Function prototype	<code>void dmamux_synchronization_init(dmamux_multiplexer_channel_enum channelx, dmamux_sync_parameter_struct *init_struct);</code>
Function descriptions	initialize DMAMUX request multiplexer channel synchronization mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum <code>dmamux_multiplexer_channel_enum</code></a> .
Input parameter{in}	
init_struct	the initialization data needed to initialize DMAMUX request multiplexer channel synchronization mode, refer to <a href="#">Table 3-290. Structure <code>dmamux_sync_parameter_struct</code></a>
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 3-327. Function `dmamux_synchronization_enable`

Function name	<code>dmamux_synchronization_enable</code>
Function prototype	<code>void dmamux_synchronization_enable(dmamux_multiplexer_channel_enum channelx);</code>
Function descriptions	enable synchronization mode
Precondition	-
The called functions	-

Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum dmamux_multiplexer_channel_enum</a> .
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 3-328. 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	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum dmamux_multiplexer_channel_enum</a> .
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 3-329. Function dmamux\_event\_generation\_enable**

Function name	dmamux_event_generation_enable
Function prototype	void dmamux_event_generation_enable(dmamux_multiplexer_channel_enum

	channelx);
<b>Function descriptions</b>	enable event generation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum dmamux multiplexer channel enum.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-330. Function dmamux\_event\_generation\_disable**

<b>Function name</b>	dmamux_event_generation_disable
<b>Function prototype</b>	void dmamux_event_generation_disable(dmamux_multiplexer_channel_enum channelx);
<b>Function descriptions</b>	disable event generation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum dmamux multiplexer channel enum.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-331. Function `dmamux_gen_struct_para_init`

Function name	<code>dmamux_gen_struct_para_init</code>
Function prototype	<code>void dmamux_gen_struct_para_init(dmamux_gen_parameter_struct *init_struct);</code>
Function descriptions	initialize the parameters of DMAMUX request generator structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
<code>init_struct</code>	the initialization data needed to initialize DMAMUX request generator channel, refer to <a href="#">Table 3-291. Structure <code>dmamux_gen_parameter_struct</code></a> .
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 3-332. Function `dmamux_request_generator_init`

Function name	<code>dmamux_request_generator_init</code>
Function prototype	<code>void dmamux_request_generator_init(dmamux_generator_channel_enum channelx, dmamux_gen_parameter_struct *init_struct);</code>
Function descriptions	initialize DMAMUX request generator channel
Precondition	-
The called functions	-
Input parameter{in}	
<code>channelx</code>	DMAMUX generation channel, refer to <a href="#">Table 3-294. Enum <code>dmamux_generator_channel_enum</code></a> .
Input parameter{in}	
<code>init_struct</code>	the initialization data needed to initialize DMAMUX request generator channel, refer to <a href="#">Table 3-291. Structure <code>dmamux_gen_parameter_struct</code></a> .
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 3-333. 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	DMAMUX generation channel, refer to <a href="#">Table 3-294. Enum dmamux_generator_channel_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMAMUX request generator channel 0 */

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 3-334. 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);
<b>Function descriptions</b>	disable DMAMUX request generator channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	DMAMUX generation channel, refer to <a href="#">Table 3-294. Enum dmamux_generator_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMAMUX request generator channel 0 */
```

```
dmamux_request_generator_channel_disable(DMAMUX_GENCH0);
```

### dmamux\_synchronization\_polarity\_config

The description of dmamux\_synchronization\_polarity\_config is shown as below:

**Table 3-335. Function dmamux\_synchronization\_polarity\_config**

<b>Function name</b>	dmamux_synchronization_polarity_config
<b>Function prototype</b>	void dmamux_synchronization_polarity_config(dmamux_multiplexer_channel_enum channelx, uint32_t polarity);
<b>Function descriptions</b>	configure synchronization input polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum dmamux_multiplexer_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>polarity</b>	synchronization input polarity
DMAMUX_SYNC_NO_EVENT	no event detection
DMAMUX_SYNC_RISING	rising edge
DMAMUX_SYNC_FALLING	falling edge
DMAMUX_SYNC_RISING_FALLING	rising and falling edges
<b>Output parameter{out}</b>	
-	-

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 3-336. Function dmamux\_request\_forward\_number\_config**

Function name	dmamux_request_forward_number_config
Function prototype	void dmamux_request_forward_number_config(dmamux_multiplexer_channel_enum channelx, uint32_t number);
Function descriptions	configure number of DMA requests to forward
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum dmamux_multiplexer_channel_enum</a> .
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 3-337. 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}	
<b>channelx</b>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum <i>dmamux_multiplexer_channel_enum</i></a> .
Input parameter{in}	
<b>id</b>	synchronization input identification
<i>DMAMUX_SYNC_EXTI0</i>	synchronization input is EXTI0
<i>DMAMUX_SYNC_EXTI1</i>	synchronization input is EXTI1
<i>DMAMUX_SYNC_EXTI2</i>	synchronization input is EXTI2
<i>DMAMUX_SYNC_EXTI3</i>	synchronization input is EXTI3
<i>DMAMUX_SYNC_EXTI4</i>	synchronization input is EXTI4
<i>DMAMUX_SYNC_EXTI5</i>	synchronization input is EXTI5
<i>DMAMUX_SYNC_EXTI6</i>	synchronization input is EXTI6
<i>DMAMUX_SYNC_EXTI7</i>	synchronization input is EXTI7
<i>DMAMUX_SYNC_EXTI8</i>	synchronization input is EXTI8
<i>DMAMUX_SYNC_EXTI9</i>	synchronization input is EXTI9
<i>DMAMUX_SYNC_EXTI10</i>	synchronization input is EXTI10
<i>DMAMUX_SYNC_EXTI11</i>	synchronization input is EXTI11
<i>DMAMUX_SYNC_EXTI12</i>	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

DMAMUX_SYNC_EVT x_OUT3	synchronization input is Evt_out3
DMAMUX_SYNC_LPTI MER_OUT	synchronization input is LPTIMER_OUT
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 3-338. Function dmamux\_request\_id\_config**

<b>Function name</b>	dmamux_request_id_config
<b>Function prototype</b>	void dmamux_request_id_config(dmamux_multiplexer_channel_enum channelx, uint32_t id);
<b>Function descriptions</b>	configure multiplexer input identification
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>channelx</b>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-293. Enum dmamux_multiplexer_channel_enum</a> .
Input parameter{in}	
<b>id</b>	input DMA request identification
DMA_REQUEST_M2 M	memory to memory transfer
DMA_REQUEST_GE NERATOR0	DMAMUX request generator 0
DMA_REQUEST_GE NERATOR1	DMAMUX request generator 1
DMA_REQUEST_GE NERATOR2	DMAMUX request generator 2
DMA_REQUEST_GE NERATOR3	DMAMUX request generator 3
DMA_REQUEST_AD C0	DMAMUX request ADC0
DMA_REQUEST_DA C0_CH0	DMAMUX request DAC0_CH0

DMA_REQUEST_DAC0_CH1	DMAMUX request DAC0_CH1
DMA_REQUEST_TIMER5_UP	request DMAMUX TIMER5_UP
DMA_REQUEST_TIMER6_UP	request DMAMUX TIMER6_UP
DMA_REQUEST_SPI0_RX	request DMAMUX SPI0_RX
DMA_REQUEST_SPI0_TX	request DMAMUX SPI0_TX
DMA_REQUEST_SPI1_RX	request DMAMUX SPI1_RX
DMA_REQUEST_SPI1_TX	request DMAMUX SPI1_TX
DMA_REQUEST_SPI2_RX	request DMAMUX SPI2_RX
DMA_REQUEST_SPI2_TX	request DMAMUX SPI2_TX
DMA_REQUEST_I2C0_RX	request DMAMUX I2C0_RX
DMA_REQUEST_I2C0_TX	request DMAMUX I2C0_TX
DMA_REQUEST_I2C1_RX	request DMAMUX I2C1_RX
DMA_REQUEST_I2C1_TX	request DMAMUX I2C1_TX
DMA_REQUEST_I2C2_RX	request DMAMUX I2C2_RX
DMA_REQUEST_I2C2_TX	request DMAMUX I2C2_TX
DMA_REQUEST_I2C3_RX	request DMAMUX I2C3_RX
DMA_REQUEST_I2C3_TX	request DMAMUX I2C3_TX
DMA_REQUEST_USART0_RX	request DMAMUX USART0_RX
DMA_REQUEST_USART0_TX	request DMAMUX USART0_TX
DMA_REQUEST_USART1_RX	request DMAMUX USART1_RX
DMA_REQUEST_USART1_TX	request DMAMUX USART1_TX
DMA_REQUEST_USART2_RX	request DMAMUX USART2_RX

ART2_RX	
DMA_REQUEST_US ART2_TX	request DMAMUX USART2_TX
DMA_REQUEST_UA RT3_RX	request DMAMUX UART3_RX
DMA_REQUEST_UA RT3_TX	request DMAMUX UART3_TX
DMA_REQUEST_UA RT4_RX	request DMAMUX UART4_RX
DMA_REQUEST_UA RT4_TX	request DMAMUX UART4_TX
DMA_REQUEST_AD C1	request DMAMUX ADC1
DMA_REQUEST_AD C2	request DMAMUX ADC2
DMA_REQUEST_AD C3	request DMAMUX ADC3
DMA_REQUEST_QS PI	request DMAMUX QSPI
DMA_REQUEST_DA C1_CH0	request DMAMUX DAC1_CH0
DMA_REQUEST_DA C1_CH1	request DMAMUX DAC1_CH1
DMA_REQUEST_TI MER0_CH0	request DMAMUX TIMER0_CH0
DMA_REQUEST_TI MER0_CH1	request DMAMUX TIMER0_CH1
DMA_REQUEST_TI MER0_CH2	request DMAMUX TIMER0_CH2
DMA_REQUEST_TI MER0_CH3	request DMAMUX TIMER0_CH3
DMA_REQUEST_TI MER0_CH0N	request DMAMUX TIMER0_CH0N
DMA_REQUEST_TI MER0_CH1N	request DMAMUX TIMER0_CH1N
DMA_REQUEST_TI MER0_CH2N	request DMAMUX TIMER0_CH2N
DMA_REQUEST_TI MER0_CH3N	request DMAMUX TIMER0_CH3N
DMA_REQUEST_TI MER0_UP	request DMAMUX TIMER0_UP
DMA_REQUEST_TI MER0_TRIG	request DMAMUX TIMER0_TRIG



DMA_REQUEST_TIMER0_COM	request DMAMUX TIMER0_COM
DMA_REQUEST_TIMER7_CH0	request DMAMUX TIMER7_CH0
DMA_REQUEST_TIMER7_CH1	request DMAMUX TIMER7_CH1
DMA_REQUEST_TIMER7_CH2	request DMAMUX TIMER7_CH2
DMA_REQUEST_TIMER7_CH3	request DMAMUX TIMER7_CH3
DMA_REQUEST_TIMER7_CH0N	request DMAMUX TIMER7_CH0N
DMA_REQUEST_TIMER7_CH1N	request DMAMUX TIMER7_CH1N
DMA_REQUEST_TIMER7_CH2N	request DMAMUX TIMER7_CH2N
DMA_REQUEST_TIMER7_CH3N	request DMAMUX TIMER7_CH3N
DMA_REQUEST_TIMER7_UP	request DMAMUX TIMER7_UP
DMA_REQUEST_TIMER7_TRIG	request DMAMUX TIMER7_TRIG
DMA_REQUEST_TIMER7_COM	request DMAMUX TIMER7_COM
DMA_REQUEST_TIMER1_CH0	request DMAMUX TIMER1_CH0
DMA_REQUEST_TIMER1_CH1	request DMAMUX TIMER1_CH1
DMA_REQUEST_TIMER1_CH2	request DMAMUX TIMER1_CH2
DMA_REQUEST_TIMER1_CH3	request DMAMUX TIMER1_CH3
DMA_REQUEST_TIMER1_UP	request DMAMUX TIMER1_UP
DMA_REQUEST_TIMER1_TRIG	request DMAMUX TIMER1_TRIG
DMA_REQUEST_TIMER2_CH0	request DMAMUX TIMER2_CH0
DMA_REQUEST_TIMER2_CH1	request DMAMUX TIMER2_CH1
DMA_REQUEST_TIMER2_CH2	request DMAMUX TIMER2_CH2
DMA_REQUEST_TIMER2_CH3	request DMAMUX TIMER2_CH3

MER2_CH3	
DMA_REQUEST_TI MER2_UP	request DMAMUX TIMER2_UP
DMA_REQUEST_TI MER2_TRIG	request DMAMUX TIMER2_TRIG
DMA_REQUEST_TI MER3_CH0	request DMAMUX TIMER3_CH0
DMA_REQUEST_TI MER3_CH1	request DMAMUX TIMER3_CH1
DMA_REQUEST_TI MER3_CH2	request DMAMUX TIMER3_CH2
DMA_REQUEST_TI MER3_CH3	request DMAMUX TIMER3_CH3
DMA_REQUEST_TI MER3_UP	request DMAMUX TIMER3_UP
DMA_REQUEST_TI MER3_TRIG	request DMAMUX TIMER3_TRIG
DMA_REQUEST_TI MER4_CH0	request DMAMUX TIMER4_CH0
DMA_REQUEST_TI MER4_CH1	request DMAMUX TIMER4_CH1
DMA_REQUEST_TI MER4_CH2	request DMAMUX TIMER4_CH2
DMA_REQUEST_TI MER4_CH3	request DMAMUX TIMER4_CH3
DMA_REQUEST_TI MER4_UP	request DMAMUX TIMER4_UP
DMA_REQUEST_TI MER4_TRIG	request DMAMUX TIMER4_TRIG
DMA_REQUEST_TI MER14_CH0	request DMAMUX TIMER14_CH0
DMA_REQUEST_TI MER14_CH1	request DMAMUX DMAMUXTIMER14_CH1
DMA_REQUEST_TI MER14_CH0N	request DMAMUX TIMER14_CH0N
DMA_REQUEST_TI MER14_UP	request DMAMUX TIMER14_UP
DMA_REQUEST_TI MER14_TRIG	request DMAMUX TIMER14_TRIG
DMA_REQUEST_TI MER14_COM	request DMAMUX TIMER14_COM
DMA_REQUEST_TI MER15_CH0	request DMAMUX TIMER15_CH0

DMA_REQUEST_TIMER15_CH0N	request DMAMUX TIMER15_CH0N
DMA_REQUEST_TIMER15_UP	request DMAMUX TIMER15_UP
DMA_REQUEST_TIMER16_CH0	request DMAMUX TIMER16_CH0
DMA_REQUEST_TIMER16_CH0N	request DMAMUX TIMER16_CH0N
DMA_REQUEST_TIMER16_UP	request DMAMUX TIMER16_UP
DMA_REQUEST_TIMER19_CH0	request DMAMUX TIMER19_CH0
DMA_REQUEST_TIMER19_CH1	request DMAMUX TIMER19_CH1
DMA_REQUEST_TIMER19_CH2	request DMAMUX TIMER19_CH2
DMA_REQUEST_TIMER19_CH3	request DMAMUX TIMER19_CH3
DMA_REQUEST_TIMER19_CH0N	request DMAMUX TIMER19_CH0N
DMA_REQUEST_TIMER19_CH1N	request DMAMUX TIMER19_CH1N
DMA_REQUEST_TIMER19_CH2N	request DMAMUX TIMER19_CH2N
DMA_REQUEST_TIMER19_CH3N	request DMAMUX TIMER19_CH3N
DMA_REQUEST_TIMER19_UP	request DMAMUX TIMER19_UP
DMA_REQUEST_TIMER19_TRIG	request DMAMUX TIMER19_TRIG
DMA_REQUEST_TIMER19_COM	request DMAMUX TIMER19_COM
DMA_REQUEST_CAU_IN	request DMAMUX CAU_IN
DMA_REQUEST_CAU_OUT	request DMAMUX CAU_OUT
DMA_REQUEST_HRTIMER_MASTER	request DMAMUX HRTIMER_MASTER
DMA_REQUEST_HRTIMER_TIMER0	request DMAMUX HRTIMER_TIMER0
DMA_REQUEST_HRTIMER_TIMER1	request DMAMUX HRTIMER_TIMER1
DMA_REQUEST_HRTIMER_TIMER2	request DMAMUX HRTIMER_TIMER2

TIMER_TIMER2	
DMA_REQUEST_HR TIMER_TIMER3	request DMAMUX HRTIMER_TIMER3
DMA_REQUEST_HR TIMER_TIMER4	request DMAMUX HRTIMER_TIMER4
DMA_REQUEST_HR TIMER_TIMER5	request DMAMUX HRTIMER_TIMER5
DMA_REQUEST_HR TIMER_TIMER6	request DMAMUX HRTIMER_TIMER6
DMA_REQUEST_HR TIMER_TIMER7	request DMAMUX HRTIMER_TIMER7
DMA_REQUEST_DA C2_CH0	request DMAMUX DAC2_CH0
DMA_REQUEST_DA C2_CH1	request DMAMUX DAC2_CH1
DMA_REQUEST_DA C3_CH0	request DMAMUX DAC3_CH0
DMA_REQUEST_DA C3_CH1	request DMAMUX DAC3_CH1
DMA_REQUEST_HP DF_FLT0	request DMAMUX HPDF_FLT0
DMA_REQUEST_HP DF_FLT1	request DMAMUX HPDF_FLT1
DMA_REQUEST_HP DF_FLT2	request DMAMUX HPDF_FLT2
DMA_REQUEST_HP DF_FLT3	request DMAMUX HPDF_FLT3
DMA_REQUEST_FA C_READ	request DMAMUX FAC_READ
DMA_REQUEST_FA C_WRITE	request DMAMUX FAC_WRITE
DMA_REQUEST_TM U_INPUT	request DMAMUX TMU_INPUT
DMA_REQUEST_TM U_OUTPUT	request DMAMUX TMU_OUTPUT
DMA_REQUEST_CA N0	request DMAMUX CAN0
DMA_REQUEST_CA N1	request DMAMUX CAN1
DMA_REQUEST_CA N2	request DMAMUX CAN2
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 3-339. 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	DMAMUX generation channel, refer to <a href="#">Table 3-294. Enum dmamux_generator_channel_enum</a> .
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 3-340. 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	DMAMUX generation channel, refer to <a href="#">Table 3-294. Enum dmamux_generator_channel_enum</a> .
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 3-341. 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	DMAMUX generation channel, refer to <a href="#">Table 3-294. Enum dmamux_generator_channel_enum</a> .
Input parameter{in}	
id	trigger input identification
DMAMUX_TRIGGER_EXTI0	trigger input is EXTI0
DMAMUX_TRIGGER_EXTI1	trigger input is EXTI1
DMAMUX_TRIGGER_	trigger input is EXTI2

<i>EXTI2</i>	
<i>DMAMUX_TRIGGER_</i> <i>EXTI3</i>	trigger input is EXTI3
<i>DMAMUX_TRIGGER_</i> <i>EXTI4</i>	trigger input is EXTI4
<i>DMAMUX_TRIGGER_</i> <i>EXTI5</i>	trigger input is EXTI5
<i>DMAMUX_TRIGGER_</i> <i>EXTI6</i>	trigger input is EXTI6
<i>DMAMUX_TRIGGER_</i> <i>EXTI7</i>	trigger input is EXTI7
<i>DMAMUX_TRIGGER_</i> <i>EXTI8</i>	trigger input is EXTI8
<i>DMAMUX_TRIGGER_</i> <i>EXTI9</i>	trigger input is EXTI9
<i>DMAMUX_TRIGGER_</i> <i>EXTI10</i>	trigger input is EXTI10
<i>DMAMUX_TRIGGER_</i> <i>EXTI11</i>	trigger input is EXTI11
<i>DMAMUX_TRIGGER_</i> <i>EXTI12</i>	trigger input is EXTI12
<i>DMAMUX_TRIGGER_</i> <i>EXTI13</i>	trigger input is EXTI13
<i>DMAMUX_TRIGGER_</i> <i>EXTI14</i>	trigger input is EXTI14
<i>DMAMUX_TRIGGER_</i> <i>EXTI15</i>	trigger input is EXTI15
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT0</i>	trigger input is Evt_out0
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT1</i>	trigger input is Evt_out1
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT2</i>	trigger input is Evt_out2
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT3</i>	trigger input is Evt_out3
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT4</i>	trigger input is Evt_out4
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT5</i>	trigger input is Evt_out5
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT6</i>	trigger input is Evt_out6
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT7</i>	trigger input is Evt_out7

<i>DMAMUX_TRIGGER_RTC_WAKEUP</i>	trigger input is wakeup
<i>DMAMUX_TRIGGER_CMP0_OUTPUT</i>	trigger input is CMP0 output
<i>DMAMUX_TRIGGER_CMP1_OUTPUT</i>	trigger input is CMP1 output
<i>DMAMUX_TRIGGER_I2C0_WAKEUP</i>	trigger input is I2C0 wakeup
<i>DMAMUX_TRIGGER_I2C1_WAKEUP</i>	trigger input is I2C1 wakeup
<i>DMAMUX_TRIGGER_I2C2_WAKEUP</i>	trigger input is I2C2 wakeup
<i>DMAMUX_TRIGGER_I2C3_WAKEUP</i>	trigger input is I2C3 wakeup
<i>DMAMUX_TRIGGER_I2C0_INT_EVENT</i>	trigger input is I2C0 interrupt event
<i>DMAMUX_TRIGGER_I2C1_INT_EVENT</i>	trigger input is I2C1 interrupt event
<i>DMAMUX_TRIGGER_I2C2_INT_EVENT</i>	trigger input is I2C2 interrupt event
<i>DMAMUX_TRIGGER_I2C3_INT_EVENT</i>	trigger input is I2C3 interrupt event
<i>DMAMUX_TRIGGER_ADC2_INT</i>	ADC2 interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-342. Function dmamux\_flag\_get**

<b>Function name</b>	dmamux_flag_get
<b>Function prototype</b>	FlagStatus dmamux_flag_get(dmamux_flag_enum flag);
<b>Function descriptions</b>	get DMAMUX flag
<b>Precondition</b>	-
<b>The called functions</b>	-



Input parameter{in}	
flag	flag type, refer to <a href="#">Table 3-296. Enum dmamux_flag_enum.</a>
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMAMUX flag */
```

```
FlagStatus flag = RESET;
```

```
flag = dmamux_flag_get(DMAMUX_FLAG_GENCH0_TO);
```

### dmamux\_flag\_clear

The description of dmamux\_flag\_clear is shown as below:

**Table 3-343. 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 <a href="#">Table 3-296. Enum dmamux_flag_enum.</a>
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 3-344. 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, refer to <a href="#">Table 3-295. Enum dmamux_interrupt_enum</a>
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 3-345. 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, refer to <a href="#">Table 3-295. Enum dmamux_interrupt_enum</a> .
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 3-346. 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 <a href="#">Table 3-297. Enum dmamux_interrupt_flag_enum.</a>
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMAMUX interrupt flag */
```

```
FlagStatus flag = RESET;
```

```
flag = dmamux_interrupt_flag_get(DMAMUX_INT_FLAG_MUXCH0_SO);
```

### dmamux\_interrupt\_flag\_clear

The description of dmamux\_interrupt\_flag\_clear is shown as below:

**Table 3-347. Function dmamux\_interrupt\_flag\_clear**

Function name	dmamux_interrupt_flag_clear
Function prototype	FlagStatus dmamux_interrupt_flag_get(dmamux_interrupt_flag_enum int_flag);
Function descriptions	clear DMAMUX interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	flag type, refer to <a href="#">Table 3-297. Enum dmamux_interrupt_flag_enum.</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMAMUX interrupt flag */
```

```
dmamux_interrupt_flag_clear(DMAMUX_INT_FLAG_MUXCH0_SO);
```

## 3.11. CLA

The configurable logic array provides 256 programmable digital logic operations for external pins, ADC and timers without intervention from the CPU. Four independent CLA units are implemented in this module. Each CLA unit supports configurable asynchronous and synchronous output for GPIO pins. The CLA registers are listed in chapter [3.11.1](#), the CLA firmware functions are introduced in chapter [3.11.2](#).

### 3.11.1. Descriptions of Peripheral registers

CLA registers are listed in the table shown as below:

**Table 3-348. CLA Registers**

Registers	Descriptions
CLA_GCTL	Global control register
CLA_INTE	Interrupt enable register
CLA_INTF	Interrupt flag register
CLA_STAT	Status register
CLAx_SIGS(x=0..3)	Signal selection register
CLAx_LCUCTL(x=0..3)	LCU control register
CLAx_CTL(x=0..3)	Control register

### 3.11.2. Descriptions of Peripheral functions

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

**Table 3-349. CLA firmware function**

Function name	Function description
cla_deinit	reset CLA
cla_enable	enable CLA
cla_disable	disable CLA
cla_output_state_get	get CLA output state
cla_sigs_input_config	configure signal selector input
cla_lcu_control_config	configure CLA LCU control register value
cla_output_config	configure CLA output
cla_output_enable	enable CLA output
cla_output_disable	disable CLA output
cla_flip_flop_output_reset	reset the flip-flop output asynchronously
cla_flip_flop_clockpolarity_config	configure clock polarity of flip-flop
cla_flip_flop_clocksource_config	configure clock source of flip-flop
cla_flag_get	check CLA flag is set or not
cla_flag_clear	clear CLA flag
cla_negedge_interrupt_enable	enable CLA falling edge interrupt
cla_negedge_interrupt_disable	disable CLA falling edge interrupt
cla_posedge_interrupt_enable	enable CLA rising edge interrupt
cla_posedge_interrupt_disable	disable CLA rising edge interrupt
cla_interrupt_flag_get	check CLA interrupt flag is set or not
cla_interrupt_flag_clear	clear CLA interrupt flag

#### cla\_deinit

The description of cla\_deinit is shown as below:

**Table 3-350. Function cla\_deinit**

<b>Function name</b>	cla_deinit
<b>Function prototype</b>	void cla_deinit(void);
<b>Function descriptions</b>	reset CLA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* CLA deinitialize */
cla_deinit ();
```

### cla\_enable

The description of cla\_enable is shown as below:

**Table 3-351. Function cla\_enable**

<b>Function name</b>	cla_enable
<b>Function prototype</b>	void cla_enable(cla_enum cla_periph);
<b>Function descriptions</b>	enable CLA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cla_periph</b>	cla periph
CLAx	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CLA0 */
cla_enable (CLA0);
```

### cla\_disable

The description of cla\_disable is shown as below:

Table 3-352. Function cla\_disable

Function name	cla_disable
Function prototype	cla_disable(cla_enum cla_periph);
Function descriptions	disable CLA
Precondition	-
The called functions	-
Input parameter{in}	
cla_periph	cla_periph
CLAx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CLA0 */
cla_disable (CAL0);
```

### cla\_output\_state\_get

The description of cla\_output\_state\_get is shown as below:

Table 3-353. Function cla\_output\_state\_get

Function name	cla_output_state_get
Function prototype	cla_outputstatus_enum cla_output_state_get(cla_enum cla_periph);
Function descriptions	get CLA output state
Precondition	-
The called functions	-
Input parameter{in}	
cla_periph	cla_periph
CLAx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
cla_outputstatus_enum	CLA_OUTPUT_HIGH / CLA_OUTPUT_LOW

Example:

```
/* get CLA0 output state */
cla_output_state_get (CLA0);
```

## cla\_sigs\_input\_config

The description of cla\_sigs\_input\_config is shown as below:

**Table 3-354. Function cla\_sigs\_input\_config**

<b>Function name</b>	cla_sigs_input_config
<b>Function prototype</b>	void cla_sigs_input_config(cla_enum cla_periph, cla_sigs_enum sigs, uint32_t input);
<b>Function descriptions</b>	configure signal selector input
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cla_periph</b>	cla periph
CLAx	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>sigs</b>	signal selector
SIGS0	signal selector 0
SIGS1	signal selector 1
<b>Input parameter{in}</b>	
<b>input</b>	input of signal selector
CLA0SIGS0_CLA0_ASYNC_OUT	CLA0SIGS0_CLA0_ASYNC_OUT
CLA1SIGS0_CLA0_ASYNC_OUT	CLA1SIGS0_CLA0_ASYNC_OUT
CLA2SIGS0_CLA0_ASYNC_OUT	CLA2SIGS0_CLA0_ASYNC_OUT
CLA3SIGS0_CLA0_ASYNC_OUT	CLA03SIGS0_CLA0_ASYNC_OUT
CLA0SIGS0_CLA1_ASYNC_OUT	CLA0SIGS0_CLA1_ASYNC_OUT
CLA1SIGS0_CLA1_ASYNC_OUT	CLA1SIGS0_CLA1_ASYNC_OUT
CLA2SIGS0_CLA1_ASYNC_OUT	CLA2SIGS0_CLA1_ASYNC_OUT
CLA0SIGS0_CLA2_ASYNC_OUT	CLA3SIGS0_CLA1_ASYNC_OUT
CLA1SIGS0_CLA2_ASYNC_OUT	CLA1SIGS0_CLA2_ASYNC_OUT
CLA2SIGS0_CLA2_ASYNC_OUT	CL21SIGS0_CLA2_ASYNC_OUT
.....	.....
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CLA0SIGS0_CLA0_ASYNC_OUT for CLA2 SIGS0 input */
cla_sigs_input_config (CLA2, SIGS0, CLA0SIGS0_CLA0_ASYNC_OUT);
```

## cla\_lcu\_control\_config

The description of cla\_lcu\_control\_config is shown as below:

Table 3-355. Function `cla_lcu_control_config`

Function name	<code>cla_lcu_control_config</code>
Function prototype	<code>void cla_lcu_control_config(cla_enum cla_periph, uint8_t lcuctl_value);</code>
Function descriptions	configure CLA LCU control register value
Precondition	-
The called functions	-
Input parameter{in}	
<code>cla_periph</code>	cla periph
<code>CLAx</code>	(x=0,1,2,3)
Input parameter{in}	
<code>lcuctl_value</code>	the value of LCU control register(0-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CLA LCU control register value */
```

```
cla_lcu_control_config (CLA0, 0xC0);
```

### `cla_output_config`

The description of `cla_output_config` is shown as below:

Table 3-356. Function `cla_output_config`

Function name	<code>cla_output_config</code>
Function prototype	<code>void cla_output_config(cla_enum cla_periph, uint32_t output);</code>
Function descriptions	configure CLA output
Precondition	-
The called functions	-
Input parameter{in}	
<code>cla_periph</code>	cla periph
<code>CLAx</code>	(x=0,1,2,3)
Input parameter{in}	
<code>output</code>	output of CLA
<code>FLIP_FLOP_OUTPUT</code>	flip-flop output is selected as CLA output
<code>LCU_RESULT</code>	LCU result is selected as CLA output
Output parameter{out}	
-	-
Return value	
-	-

Example:



```
/* configure LCU_RESULT for CLA0 output */
```

```
cla_output_config (CLA0, LCU_RESULT);
```

### cla\_output\_enable

The description of cla\_output\_enable is shown as below:

**Table 3-357. Function cla\_output\_enable**

<b>Function name</b>	cla_output_enable
<b>Function prototype</b>	void cla_output_enable (cla_enum cla_periph);
<b>Function descriptions</b>	enable CLA output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cla_periph</b>	cla periph
CLAx	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CLA0 output */
```

```
void cla_output_enable (CLA0);
```

### cla\_output\_disable

The description of cla\_output\_disable is shown as below:

**Table 3-358. Function cla\_output\_disable**

<b>Function name</b>	cla_output_disable
<b>Function prototype</b>	void cla_output_disable (cla_enum cla_periph);
<b>Function descriptions</b>	disable CLA output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cla_periph</b>	cla periph
CLAx	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CLA0 output */
```

```
void cla_output_disable (CLA0);
```

### cla\_flip\_flop\_output\_reset

The description of cla\_flip\_flop\_output\_reset is shown as below:

**Table 3-359. Function cla\_flip\_flop\_output\_reset**

<b>Function name</b>	cla_flip_flop_output_reset
<b>Function prototype</b>	void cla_flip_flop_output_reset(cla_enum cla_periph);
<b>Function descriptions</b>	reset the flip-flop output asynchronously
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cla_periph</b>	cla periph
<b>CLAx</b>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset the CLA0 flip-flop output asynchronously */
```

```
cla_flip_flop_output_reset (CLA0);
```

### cla\_flip\_flop\_clockpolarity\_config

The description of cla\_flip\_flop\_clockpolarity\_config is shown as below:

**Table 3-360. Function cla\_flip\_flop\_clockpolarity\_config**

<b>Function name</b>	cla_flip_flop_clockpolarity_config
<b>Function prototype</b>	void cla_flip_flop_clockpolarity_config(cla_enum cla_periph, uint32_t polarity);
<b>Function descriptions</b>	configure clock polarity of flip-flop
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cla_periph</b>	cla periph
<b>CLAx</b>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>polarity</b>	clock polarity of flip-flop
<b>CLA_CLOCKPOLARITY_POSEDGE</b>	clock rising edge is valid

<i>CLA_CLOCKPOLARITY_NEGEDGE</i>	clock falling edge is valid
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure clock rising edge is valid for CLA0 flip-flop */
```

```
cla_flip_flop_clockpolarity_config (CLA0, CLA_CLOCKPOLARITY_POSEDGE);
```

### cla\_flip\_flop\_clocksource\_config

The description of cla\_flip\_flop\_clocksource\_config is shown as below:

**Table 3-361. Function cla\_flip\_flop\_clocksource\_config**

<b>Function name</b>	cla_flip_flop_clocksource_config
<b>Function prototype</b>	void cla_flip_flop_clocksource_config(cla_enum cla_periph, uint32_t clock_source);
<b>Function descriptions</b>	configure clock source of flip-flop
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cla_periph</b>	cla periph
<i>CLAx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>clock_source</b>	clock source of flip-flop
<i>PRE_CLA_LCU_RESULT</i>	the result of the previous CLA units
<i>SIGS0_OUTPUT</i>	the signal selector output of SIGS0
<i>HCLK</i>	The HCLK
<i>TIMER_TRGO</i>	TIMER TRGO
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the SIGS0_OUTPUT for CAL0 flip-flop's clock polarity */
```

```
cla_flip_flop_clocksource_config (CLA0, SIGS0_OUTPUT);
```

### cla\_flag\_get

The description of cla\_flag\_get is shown as below:

Table 3-362. Function `cla_flag_get`

<b>Function name</b>	<code>cla_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus cla_flag_get(cla_flag_enum flag);</code>
<b>Function descriptions</b>	check CLA flag is set or not
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	CLA flags
<code>CLA_FLAG_CLA0NF</code>	CLA0 unit falling edge flag
<code>CLA_FLAG_CLA0PF</code>	CLA0 unit rising edge flag
<code>CLA_FLAG_CLA1NF</code>	CLA1 unit falling edge flag
<code>CLA_FLAG_CLA1PF</code>	CLA1 unit falling edge flag
<code>CLA_FLAG_CLA2NF</code>	CLA2 unit falling edge flag
<code>CLA_FLAG_CLA2PF</code>	CLA2 unit falling edge flag
<code>CLA_FLAG_CLA3NF</code>	CLA3 unit falling edge flag
<code>CLA_FLAG_CLA3PF</code>	CLA3 unit falling edge flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET / RESET

Example:

```
/* check CLA0 unit falling edge flag */
```

```
cla_flag_get (CLA_FLAG_CLA0NF);
```

### **`cla_flag_clear`**

The description of `cla_flag_clear` is shown as below:

Table 3-363. Function `cla_flag_clear`

<b>Function name</b>	<code>cla_flag_clear</code>
<b>Function prototype</b>	<code>void cla_flag_clear(cla_flag_enum flag);</code>
<b>Function descriptions</b>	clear CLA flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	CLA flags
<code>CLA_FLAG_CLA0NF</code>	CLA0 unit falling edge flag
<code>CLA_FLAG_CLA0PF</code>	CLA0 unit rising edge flag
<code>CLA_FLAG_CLA1NF</code>	CLA1 unit falling edge flag
<code>CLA_FLAG_CLA1PF</code>	CLA1 unit rising edge flag
<code>CLA_FLAG_CLA2NF</code>	CLA2 unit falling edge flag
<code>CLA_FLAG_CLA2PF</code>	CLA2 unit rising edge flag

<i>CLA_FLAG_CLA3NF</i>	CLA3 unit falling edge flag
<i>CLA_FLAG_CLA3PF</i>	CLA3 unit rising edge flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear CLA0 unit falling edge flag */
cla_flag_clear (CLA_FLAG_CLA0NF);
```

### **cla\_negedge\_interrupt\_enable**

The description of `cla_negedge_interrupt_enable` is shown as below:

**Table 3-364. Function `cla_negedge_interrupt_enable`**

<b>Function name</b>	<code>cla_negedge_interrupt_enable</code>
<b>Function prototype</b>	<code>void cla_negedge_interrupt_enable(uint32_t clanie);</code>
<b>Function descriptions</b>	enable CLA falling edge interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clanie</b>	CLA falling edge interrupt
<i>CLA0NIE</i>	CLA0 falling edge interrupt enable
<i>CLA1NIE</i>	CLA1 falling edge interrupt enable
<i>CLA2NIE</i>	CLA2 falling edge interrupt enable
<i>CLA3NIE</i>	CLA3 falling edge interrupt enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CLA0 negedge interrupt */
cla_negedge_interrupt_enable (CLA0NIE);
```

### **cla\_negedge\_interrupt\_disable**

The description of `cla_negedge_interrupt_disable` is shown as below:

**Table 3-365. Function `cla_negedge_interrupt_disable`**

<b>Function name</b>	<code>cla_negedge_interrupt_disable</code>
<b>Function prototype</b>	<code>void cla_negedge_interrupt_disable(uint32_t clanidis);</code>

<b>Function descriptions</b>	disable CLA falling edge interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clanidis</b>	CLA falling edge interrupt
<i>CLA0NI_DISABLE</i>	CLA0 falling edge interrupt disable
<i>CLA1NI_DISABLE</i>	CLA1 falling edge interrupt disable
<i>CLA2NI_DISABLE</i>	CLA2 falling edge interrupt disable
<i>CLA3NI_DISABLE</i>	CLA3 falling edge interrupt disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CLA0 negedge interrupt */
```

```
cla_negedge_interrupt_disable (CLA0NI_DISABLE);
```

### cla\_posedge\_interrupt\_enable

The description of cla\_posedge\_interrupt\_enable is shown as below:

**Table 3-366. Function cla\_posedge\_interrupt\_enable**

<b>Function name</b>	cla_posedge_interrupt_enable
<b>Function prototype</b>	void cla_posedge_interrupt_enable(uint32_t clapie);
<b>Function descriptions</b>	enable CLA rising edge interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clapie</b>	CLA rising edge interrupt
<i>CLA0PIE</i>	CLA0 rising edge interrupt enable
<i>CLA1PIE</i>	CLA1 rising edge interrupt enable
<i>CLA2PIE</i>	CLA2 rising edge interrupt enable
<i>CLA3NIE</i>	CLA3 rising edge interrupt enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CLA0 rising edge interrupt*/
```

```
cla_posedge_interrupt_enable (CLA0PIE);
```

## cla\_posedge\_interrupt\_disable

The description of cla\_posedge\_interrupt\_disable is shown as below:

**Table 3-367. Function cla\_posedge\_interrupt\_disable**

<b>Function name</b>	cla_posedge_interrupt_disable
<b>Function prototype</b>	void cla_posedge_interrupt_disable(uint32_t clapidis);
<b>Function descriptions</b>	disable CLA rising edge interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clapidis</b>	CLA rising edge interrupt
<i>CLA0PI_DISABLE</i>	CLA0 rising edge interrupt disable
<i>CLA1PI_DISABLE</i>	CLA1 rising edge interrupt disable
<i>CLA2PI_DISABLE</i>	CLA2 rising edge interrupt disable
<i>CLA3PI_DISABLE</i>	CLA3 rising edge interrupt disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CLA0 rising edge interrupt */
```

```
cla_posedge_interrupt_disable (CLA0PI_DISABLE);
```

## cla\_interrupt\_flag\_get

The description of cla\_interrupt\_flag\_get is shown as below:

**Table 3-368. Function cla\_interrupt\_flag\_get**

<b>Function name</b>	cla_interrupt_flag_get
<b>Function prototype</b>	FlagStatus cla_interrupt_flag_get(cla_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	check CLA interrupt flag is set or not
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	CLA interrupt flags
<i>CLA_INT_FLAG_CLA0NF</i>	CLA0 unit falling edge interrupt flag
<i>CLA_INT_FLAG_CLA0PF</i>	CLA0 unit rising edge interrupt flag
<i>CLA_INT_FLAG_CLA1NF</i>	CLA1 unit falling edge interrupt flag
<i>CLA_INT_FLAG_CLA1PF</i>	CLA1 unit rising edge interrupt flag
<i>CLA_INTFLAG_CLA2NF</i>	CLA2 unit falling edge interrupt flag
<i>CLA_INTFLAG_CLA2PF</i>	CLA2 unit rising edge interrupt flag
<i>CLA_INTFLAG_CLA3NF</i>	CLA3 unit falling edge interrupt flag

<i>CLA_INTFLAG_CLA3PF</i>	CLA3 unit rising edge interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET / RESET

Example:

```
/* check CLA0 unit falling edge interrupt flag */
```

```
cla_interrupt_flag_get (CLA_INT_FLAG_CLA0NF);
```

### cla\_interrupt\_flag\_clear

The description of cla\_interrupt\_flag\_clear is shown as below:

**Table 3-369. Function cla\_interrupt\_flag\_clear**

<b>Function name</b>	cla_interrupt_flag_clear
<b>Function prototype</b>	void cla_interrupt_flag_clear(cla_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	clear CLA interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	CLA interrupt flags
<i>CLA_INT_FLAG_CLA0NF</i>	CLA0 unit falling edge interrupt flag
<i>CLA_INT_FLAG_CLA0PF</i>	CLA0 unit rising edge interrupt flag
<i>CLA_INT_FLAG_CLA1NF</i>	CLA1 unit falling edge interrupt flag
<i>CLA_INT_FLAG_CLA1PF</i>	CLA1 unit rising edge interrupt flag
<i>CLA_INTFLAG_CLA2NF</i>	CLA2 unit falling edge interrupt flag
<i>CLA_INTFLAG_CLA2PF</i>	CLA2 unit rising edge interrupt flag
<i>CLA_INTFLAG_CLA3NF</i>	CLA3 unit falling edge interrupt flag
<i>CLA_INTFLAG_CLA3PF</i>	CLA3 unit rising edge interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear CLA0 unit falling edge interrupt flag */
```

```
cla_interrupt_flag_clear (CLA_INT_FLAG_CLA0NF);
```



## 3.12. EXMC

The external memory controller EXMC, is used as a translator for MCU to access a variety of external memory. The EXMC registers are listed in chapter [3.12.1](#), the EXMC firmware functions are introduced in chapter [3.12.2](#).

### 3.12.1. Descriptions of Peripheral registers

EXMC registers are listed in the table shown as below:

**Table 3-370. EXMC Registers**

Registers	Descriptions
EXMC_SNCTL	SRAM/NOR flash control registers
EXMC_SNTCFG	SRAM/NOR flash timing configuration registers
EXMC_SNWTCFG	SRAM/NOR flash write timing configuration registers
EXMC_SNSTAT	SRAM/NOR flash status register
EXMC_SNLATDEC	SRAM/NOR flash data latency decrease registers

### 3.12.2. Descriptions of Peripheral functions

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

**Table 3-371. EXMC firmware function**

Function name	Function description
exmc_norsram_deinit	deinitialize EXMC NOR/DRAM regionx
exmc_norsram_struct_para_init	initialize exmc_norsram_parameter_struct with the default values
exmc_norsram_init	initialize EXMC NOR/DRAM regionx
exmc_norsram_enable	enable EXMC NOR/PSRAM regionx
exmc_norsram_disable	disable EXMC NOR/PSRAM regionx
exmc_norsram_page_size_config	configure EXMC page size
exmc_norsram_consecutive_clock_config	configure consecutive clock
exmc_norsram_write_fifo_config	configure EXMC NOR/DRAM(bank0 region0) write FIFO
exmc_norsram_status_get	get EXMC NOR/DRAM(bank0 region0) region FIFO status

#### Structure exmc\_norsram\_timing\_parameter\_struct

**Table 3-372. Structure exmc\_norsram\_timing\_parameter\_struct**

Member name	Function description
asyn_access_mode	asynchronous access mode
syn_data_latency	configure the data latency
syn_data_latency_d	configure the data latency decreasing value

Member name	Function description
ec	
syn_clk_division	configure the clock divide ratio
bus_latency	configure the bus latency
asyn_data_setup_time	configure the data setup time,asynchronous access mode valid
asyn_address_hold_time	configure the address hold time,asynchronous access mode valid
asyn_address_setup_time	configure the data setup time,asynchronous access mode valid

### Structure exmc\_norsram\_parameter\_struct

**Table 3-373. Structure exmc\_norsram\_parameter\_struct**

Member name	Function description
norsram_region	select the region of EXMC NOR/SRAM
write_mode	the write mode, synchronous mode or asynchronous mode
extended_mode	enable or disable the extended mode
asyn_wait	enable or disable the asynchronous wait function
nwait_signal	enable or disable the NWAIT signal while in synchronous bust mode
memory_write	enable or disable the write operation
nwait_config	NWAIT signal configuration
nwait_polarity	specifies the polarity of NWAIT signal from memory
burst_mode	enable or disable the burst mode
databus_width	specifies the databus width of external memory
memory_type	specifies the type of external memory
address_data_mux	specifies whether the data bus and address bus are multiplexed
read_write_timing	timing parameters for read and write if the extended mode is not used or the timing parameters for read if the extended mode is used
write_timing	timing parameters for write when the extended mode is used

### exmc\_norsram\_deinit

The description of exmc\_norsram\_deinit is shown as below:

**Table 3-374. Function exmc\_norsram\_deinit**

Function name	exmc_norsram_deinit
Function prototype	void exmc_norsram_deinit(uint32_t exmc_norsram_region);
Function descriptions	deinitialize EXMC NOR/SRAM region
Precondition	-
The called functions	-
Input parameter{in}	
exmc_norsram_region	EXMC NOR/SRAM region

EXMC_BANK0_NORS RAM_REGIONx	x=0,1,2,3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize the EXMC NOR/SRAM region1 of bank0 */
```

```
exmc_norsram_deinit(EXMC_BANK0_NORSRAM_REGION1);
```

### exmc\_norsram\_struct\_para\_init

The description of exmc\_norsram\_struct\_para\_init is shown as below:

**Table 3-375. Function exmc\_norsram\_struct\_para\_init**

Function name	exmc_norsram_struct_para_init
Function prototype	void exmc_norsram_struct_para_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
Function descriptions	initialize the struct exmc_norsram_parameter_struct
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
exmc_norsram_init_struct	Structure for initialization, the structure members can refer to <a href="#">Table 3-373. Structure exmc_norsram_parameter_struct</a>
Return value	
-	-

Example:

```
/* initialize the struct nor_init_struct */
```

```
exmc_norsram_parameter_struct nor_init_struct;
```

```
exmc_norsram_struct_para_init(&nor_init_struct);
```

### exmc\_norsram\_init

The description of exmc\_norsram\_init is shown as below:

**Table 3-376. Function exmc\_norsram\_init**

Function name	exmc_norsram_init
Function prototype	void exmc_norsram_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);

<b>Function descriptions</b>	initialize EXMC NOR/SRAM region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>exmc_norsram_init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-373. Structure exmc_norsram_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

```

/* initialize EXMC NOR/SRAM bank */

exmc_norsram_parameter_struct lcd_init_struct;

exmc_norsram_timing_parameter_struct lcd_timing_init_struct;

exmc_norsram_struct_para_init (&lcd_init_struct);

/* configure timing parameter */

lcd_timing_init_struct.asyn_access_mode = EXMC_ACCESS_MODE_A;

lcd_timing_init_struct.syn_data_latency = EXMC_DATA_LAT_2_CLK;

lcd_timing_init_struct.syn_data_latency_dec = 0;

lcd_timing_init_struct.syn_clk_division = EXMC_SYN_CLOCK_RATIO_DISABLE;

lcd_timing_init_struct.bus_latency = 1;

lcd_timing_init_struct.asyn_data_setup_time = 18;

lcd_timing_init_struct.asyn_address_hold_time = 3;

lcd_timing_init_struct.asyn_address_setup_time = 8;

lcd_timing_init_struct.byte_lane_setup_time = 0;

/* configure EXMC bus parameters */

lcd_init_struct.norsram_region = EXMC_BANK0_NORSRAM_REGION0;

lcd_init_struct.write_mode = EXMC_ASYN_WRITE;

lcd_init_struct.extended_mode = DISABLE;

lcd_init_struct.asyn_wait = DISABLE;

lcd_init_struct.nwait_signal = DISABLE;

lcd_init_struct.memory_write = ENABLE;

```

```

lcd_init_struct.nwait_config = EXMC_NWAIT_CONFIG_BEFORE;

lcd_init_struct.wrap_burst_mode = DISABLE;

lcd_init_struct.nwait_polarity = EXMC_NWAIT_POLARITY_LOW;

lcd_init_struct.burst_mode = DISABLE;

lcd_init_struct.databus_width = EXMC_NOR_DATABUS_WIDTH_16B;

lcd_init_struct.memory_type = EXMC_MEMORY_TYPE_SRAM;

lcd_init_struct.address_data_mux = DISABLE;

lcd_init_struct.read_write_timing = &lcd_timing_init_struct;

lcd_init_struct.write_timing = &lcd_timing_init_struct;

exmc_norsram_init(&lcd_init_struct);

```

### exmc\_norsram\_enable

The description of exmc\_norsram\_enable is shown as below:

**Table 3-377. Function exmc\_norsram\_enable**

Function name	exmc_norsram_enable
Function prototype	void exmc_norsram_enable(uint32_t exmc_norsram_region);
Function descriptions	enable EXMC NOR/PSRAM region
Precondition	-
The called functions	-
Input parameter{in}	
exmc_norsram_region	EXMC NOR/SRAM region
EXMC_BANK0_NORSRAM_REGIONx	x=0,1,2,3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable the EXMC NOR/SRAM region1 of bank0 */

exmc_norsram_enable(EXMC_BANK0_NORSRAM_REGION1);

```

### exmc\_norsram\_disable

The description of exmc\_norsram\_disable is shown as below:

Table 3-378. Function `exmc_norsram_disable`

<b>Function name</b>	<code>exmc_norsram_disable</code>
<b>Function prototype</b>	<code>void exmc_norsram_disable(uint32_t exmc_norsram_region);</code>
<b>Function descriptions</b>	disable EXMC NOR/PSRAM region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_norsram_region</b>	EXMC NOR/SRAM region
<i>EXMC_BANK0_NORSRAM_REGIONx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the EXMC NOR/SRAM region1 of bank0 */
exmc_norsram_disable(EXMC_BANK0_NORSRAM_REGION1);
```

### `exmc_norsram_page_size_config`

The description of `exmc_norsram_page_size_config` is shown as below:

Table 3-379. Function `exmc_norsram_page_size_config`

<b>Function name</b>	<code>exmc_norsram_page_size_config</code>
<b>Function prototype</b>	<code>void exmc_norsram_page_size_config(uint32_t page_size);</code>
<b>Function descriptions</b>	configure CRAM page size
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>page_size</b>	CRAM page size
<i>EXMC_CRAM_AUTO_SPLIT</i>	the clock is generated only during synchronous access
<i>EXMC_CRAM_PAGE_SIZE_128_BYTES</i>	page size is 128 bytes
<i>EXMC_CRAM_PAGE_SIZE_256_BYTES</i>	page size is 256 bytes
<i>EXMC_CRAM_PAGE_SIZE_512_BYTES</i>	page size is 512 bytes
<i>EXMC_CRAM_PAGE_SIZE_1024_BYTES</i>	page size is 1024 bytes
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure CRAM page size */
```

```
exmc_norsram_page_size_config (EXMC_CRAM_PAGE_SIZE_128_BYTES);
```

### exmc\_norsram\_consecutive\_clock\_config

The description of exmc\_norsram\_consecutive\_clock\_config is shown as below:

**Table 3-380. Function exmc\_norsram\_consecutive\_clock\_config**

<b>Function name</b>	exmc_norsram_consecutive_clock_config
<b>Function prototype</b>	void exmc_norsram_consecutive_clock_config(uint32_t clock_mode);
<b>Function descriptions</b>	configure consecutive clock mode (consecutive clock is only supported in EXMC BANK0 REGION0)
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clock_mode</b>	the mode of consecutive clock
EXMC_CLOCK_SYN_MODE	the clock is generated only during synchronous access
EXMC_CLOCK_UNCONDITIONALLY	the clock is generated unconditionally
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure consecutive clock */
```

```
exmc_norsram_consecutive_clock_config(EXMC_CLOCK_SYN_MODE);
```

### exmc\_norsram\_write\_fifo\_config

The description of exmc\_norsram\_write\_fifo\_config is shown as below:

**Table 3-381. Function exmc\_norsram\_write\_fifo\_config**

<b>Function name</b>	exmc_norsram_write_fifo_config
<b>Function prototype</b>	void exmc_norsram_write_fifo_config(uint32_t wr_fifo);
<b>Function descriptions</b>	configure EXMC NOR/SRAM(bank0 region0) write FIFO
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>wr_fifo</b>	write FIFO configuration
<i>EXMC_WRITE_FIFO_ENABLE</i>	enable write FIFO
<i>EXMC_WRITE_FIFO_DISABLE</i>	disable write FIFO
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the write FIFO function */
exmc_norsram_write_fifo_config(EXMC_WRITE_FIFO_ENABLE);
```

### exmc\_norsram\_status\_get

The description of exmc\_norsram\_status\_get is shown as below:

**Table 3-382. Function exmc\_norsram\_status\_get**

<b>Function name</b>	exmc_norsram_status_get
<b>Function prototype</b>	uint32_t exmc_norsram_status_get(void);
<b>Function descriptions</b>	get EXMC NOR/SRAM(bank0 region0) region FIFO status
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<b>uint32_t</b>	NOR/SRAM write FIFO status
<i>EXMC_FIFO_NOT_EMPTY</i>	NOR/SRAM write FIFO is not empty
<i>EXMC_FIFO_EMPTY</i>	NOR/SRAM write FIFO is empty

Example:

```
/* get the EXMC NOR/SRAM write FIFO status */
uint32_t status;
status = exmc_norsram_status_get();
```



## 3.13. EXTI

EXTI is the interrupt/event controller in the MCU. It contains up to 39 independent edge detectors and generates interrupt requests or events to the processor. The EXTI registers are listed in chapter [3.13.1](#), the EXTI firmware functions are introduced in chapter [3.13.2](#).

### 3.13.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

**Table 3-383. EXTI Registers**

Registers	Descriptions
EXTI_INTEN0	interrupt enable register 0
EXTI_EVEN0	event enable register 0
EXTI_RTEN0	rising edge trigger enable register 0
EXTI_FTEN0	falling edge trigger enable register 0
EXTI_SWIEV0	software interrupt event register 0
EXTI_PD0	pending register 0
EXTI_INTEN1	interrupt enable register 1
EXTI_EVEN1	event enable register 1
EXTI_RTEN1	rising edge trigger enable register 1
EXTI_FTEN1	falling edge trigger enable register 1
EXTI_SWIEV1	software interrupt event register 1
EXTI_PD1	pending register 1

### 3.13.2. Descriptions of Peripheral functions

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

**Table 3-384. 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
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 3-385. Enum 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
EXTI_25	EXTI line 25
EXTI_26	EXTI line 26
EXTI_27	EXTI line 27
EXTI_28	EXTI line 28
EXTI_29	EXTI line 29
EXTI_30	EXTI line 30
EXTI_31	EXTI line 31
EXTI_32	EXTI line 32
EXTI_33	EXTI line 33
EXTI_34	EXTI line 34
EXTI_35	EXTI line 35
EXTI_36	EXTI line 36
EXTI_37	EXTI line 37

enum name	Function description
EXTI_38	EXTI line 38

### Enum exti\_mode\_enum

Table 3-386. Enum exti\_mode\_enum

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

### Enum exti\_trig\_type\_enum

Table 3-387. Enum 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 3-388. 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 3-389. 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 <a href="#">Table 3-385. Enum exti_line_enum</a>
Input parameter{in}	
mode	EXTI mode, refer to <a href="#">Table 3-386. Enum exti_mode_enum</a>
Input parameter{in}	
trig_type	trigger type, refer to <a href="#">Table 3-387. Enum exti_trig_type_enum</a>
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 3-390. 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 <a href="#">Table 3-385. Enum exti_line_enum</a>
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 3-391. Function exti\_interrupt\_disable**

<b>Function name</b>	exti_interrupt_disable
<b>Function prototype</b>	void exti_interrupt_disable(exti_line_enum linex);
<b>Function descriptions</b>	disable the interrupts from EXTI line x
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-392. Function exti\_event\_enable**

<b>Function name</b>	exti_event_enable
<b>Function prototype</b>	void exti_event_enable(exti_line_enum linex);
<b>Function descriptions</b>	enable the events from EXTI line x
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-393. Function exti\_event\_disable**

<b>Function name</b>	exti_event_disable
<b>Function prototype</b>	void exti_event_disable(exti_line_enum linex);
<b>Function descriptions</b>	disable the events from EXTI line x
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-394. Function exti\_software\_interrupt\_enable**

<b>Function name</b>	exti_software_interrupt_enable
<b>Function prototype</b>	void exti_software_interrupt_enable(exti_line_enum linex);
<b>Function descriptions</b>	enable the software interrupt event from EXTI line x
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-395. Function exti\_software\_interrupt\_disable**

<b>Function name</b>	exti_software_interrupt_disable
<b>Function prototype</b>	void exti_software_interrupt_disable(exti_line_enum linex);
<b>Function descriptions</b>	disable the software interrupt event from EXTI line x
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-396. Function exti\_flag\_get**

<b>Function name</b>	exti_flag_get
<b>Function prototype</b>	FlagStatus exti_flag_get(exti_line_enum linex);
<b>Function descriptions</b>	get EXTI line x interrupt pending flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-397. Function exti\_flag\_clear**

<b>Function name</b>	exti_flag_clear
<b>Function prototype</b>	void exti_flag_clear(exti_line_enum linex);
<b>Function descriptions</b>	clear EXTI line x interrupt pending flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-398. Function exti\_interrupt\_flag\_get**

<b>Function name</b>	exti_interrupt_flag_get
<b>Function prototype</b>	FlagStatus exti_interrupt_flag_get(exti_line_enum linex);
<b>Function descriptions</b>	get EXTI line x interrupt pending flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-399. Function exti\_interrupt\_flag\_clear**

<b>Function name</b>	exti_interrupt_flag_clear
<b>Function prototype</b>	void exti_interrupt_flag_clear(exti_line_enum linex);
<b>Function descriptions</b>	clear EXTI line x interrupt pending flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-385. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
exti_interrupt_flag_clear(EXTI_0);
```

## 3.14. FAC

The filter arithmetic accelerator unit can realize finite impulse response (FIR) filters and infinite impulse response (IIR) filters. The FAC registers are listed in chapter [3.14.1](#), the FAC firmware functions are introduced in chapter [3.14.2](#).

### 3.14.1. Descriptions of Peripheral registers

FAC registers are listed in the table shown as below:

**Table 3-400. FAC Registers**

Registers	Descriptions
FAC_X0BCFG	FAC X0 buffer configure register
FAC_X1BCFG	FAC X1 buffer configure register
FAC_YBCFG	FAC Y buffer configure register
FAC_PARACFG	FAC Parameter configure register
FAC_CTL	FAC control register
FAC_STAT	FAC status register
FAC_WDATA	FAC write data register
FAC_RDATA	FAC read data register

### 3.14.2. Descriptions of Peripheral functions

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

**Table 3-401. FAC firmware function**

Function name	Function description
fac_deinit	reset the FAC peripheral
fac_struct_para_init	initialize the FAC filter parameter struct with the default values
fac_fixed_data_preload_init	initialize the FAC fixed data preload parameter struct with the default values
fac_float_data_preload_init	initialize the FAC float data preload parameter struct with the default values
fac_init	initialize the FAC peripheral
fac_fixed_buffer_preload	FAC preload X0 X1 Y fixed buffer
fac_float_buffer_preload	FAC preload X0 X1 Y float buffer
fac_fixed_data_preload	FAC preload fixed data
fac_float_data_preload	FAC preload float data
fac_reset	FAC reset write and read pointers
fac_clip_config	config the FAC clip feature
fac_float_enable	enable FAC float point format
fac_float_disable	disable FAC float point format
fac_dma_enable	enable the FAC dma
fac_dma_disable	disable the FAC dma
fac_x0_config	FAC config input buffer
fac_x1_config	FAC config coefficient buffer
fac_y_config	FAC config output buffer
fac_function_config	FAC config function execute
fac_start	start the FAC
fac_stop	stop the FAC
fac_finish_calculate	finish the filter calculate
fac_interrupt_enable	enable the FAC interrupt
fac_interrupt_disable	disable the FAC interrupt
fac_interrupt_flag_get	get the FAC interrupt flag status
fac_flag_get	get the FAC flag status
fac_fixed_data_write	FAC write data with fixed point format
fac_fixed_data_read	FAC read data with fixed point format

Function name	Function description
fac_float_data_write	FAC write data with float point format
fac_float_data_read	FAC read data with fixed point format

### Structure fac\_parameter\_struct

**Table 3-402. Structure fac\_parameter\_struct**

Member name	Function description
input_addr	base address of the input buffer (X0)
input_size	size of input buffer
coeff_addr	base address of the coefficient buffer (X1)
coeff_size	size of coefficient buffer
output_addr	base address of the output buffer (Y)
output_size	size of output buffer
ipp	vector IPP length
ipq	vector IPQ length
ipr	vector IPR length
input_threshold	threshold of input buffer full
output_threshold	threshold of output buffer empty
clip	enable or disable the clipping feature
func	FAC functions select

### Structure fac\_fixed\_data\_preload\_struct

**Table 3-403. Structure fac\_fixed\_data\_preload\_struct**

Member name	Function description
coeffa_size	size of the coefficient vector A
*coeffa_ctx	context of the coefficient vector A(int16_t format)
coeffb_size	size of the coefficient vector B
*coeffb_ctx	context of the coefficient vector B(int16_t format)
input_size	size of the input data
*input_ctx	context of the input data(int16_t format)
output_size	size of the output data
*output_ctx	context of the output data(int16_t format)

### Structure fac\_float\_data\_preload\_struct

**Table 3-404. Structure fac\_float\_data\_preload\_struct**

Member name	Function description
coeffa_size	size of the coefficient vector A
*coeffa_ctx	context of the coefficient vector A(float format)
coeffb_size	size of the coefficient vector B
*coeffb_ctx	context of the coefficient vector B(float format)
input_size	size of the input data

Member name	Function description
*input_ctx	context of the input data(float format)
output_size	size of the output data
*output_ctx	context of the output data(float format)

## fac\_deinit

The description of fac\_deinit is shown as below:

**Table 3-405. Function fac\_deinit**

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

Example:

```
/* deinitialize FAC */
fac_deinit();
```

## fac\_struct\_para\_init

The description of fac\_struct\_para\_init is shown as below:

**Table 3-406. Function fac\_struct\_para\_init**

Function name	fac_struct_para_init
Function prototype	void fac_struct_para_init(fac_parameter_struct* fac_parameter);
Function descriptions	initialize the FAC filter parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
fac_parameter	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_parameter_struct facconfig;
```

```
fac_struct_para_init(&facconfig);
```

### fac\_fixed\_data\_preload\_init

The description of fac\_fixed\_data\_preload\_init is shown as below:

**Table 3-407. Function fac\_fixed\_data\_preload\_init**

Function name	fac_fixed_data_preload_init
Function prototype	void fac_fixed_data_preload_init(fac_fixed_data_preload_struct *init_struct);
Function descriptions	initialize the FAC fixed data preload parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	FAC fixed data preload init parameter struct, the structure members can refer to <a href="#">Structure fac fixed data preload struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_fixed_data_preload_struct init_struct;
```

```
fac_fixed_data_preload_init (&init_struct);
```

### fac\_float\_data\_preload\_init

The description of fac\_float\_data\_preload\_init is shown as below:

**Table 3-408. Function fac\_float\_data\_preload\_init**

Function name	fac_float_data_preload_init
Function prototype	void fac_float_data_preload_init(fac_float_data_preload_struct *init_struct);
Function descriptions	initialize the FAC float data preload parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	FAC float data preload init parameter struct, the structure members can refer to <a href="#">Structure fac float data preload struct</a>
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_float_data_preload_struct init_struct;
```

```
fac_float_data_preload_init (&init_struct);
```

## fac\_init

The description of fac\_init is shown as below:

**Table 3-409. Function fac\_init**

<b>Function name</b>	fac_init
<b>Function prototype</b>	void fac_init(fac_parameter_struct* fac_parameter);
<b>Function descriptions</b>	Initialize the FAC peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fac_parameter</b>	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* Initialize the FAC peripheral */
```

```
fac_parameter_struct facconfig;
```

```
fac_init(&facconfig);
```

## fac\_fixed\_buffer\_preload

The description of fac\_fixed\_buffer\_preload is shown as below:

**Table 3-410. Function fac\_preload**

<b>Function name</b>	fac_fixed_buffer_preload
<b>Function prototype</b>	void fac_fixed_buffer_preload(fac_fixed_data_preload_struct* init_struct);
<b>Function descriptions</b>	FAC preload X0 X1 Y fixed buffer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>init_struct</b>	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_fixed_data_preload_struct</a> .

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC preload X0 X1 Y fixed buffer */
fac_fixed_data_preload_struct faccoeff;
fac_fixed_buffer_preload (&faccoeff);
```

### fac\_float\_buffer\_preload

The description of fac\_float\_buffer\_preload is shown as below:

**Table 3-411. Function fac\_float\_buffer\_preload**

Function name	fac_float_buffer_preload
Function prototype	void fac_float_buffer_preload(fac_float_data_preload_struct* init_struct);
Function descriptions	FAC preload X0 X1 Y float buffer
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_float_data_preload_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC preload X0 X1 Y float buffer */
fac_float_data_preload_struct faccoeff;
fac_float_buffer_preload(&faccoeff);
```

### fac\_fixed\_data\_preload

The description of fac\_fixed\_data\_preload is shown as below:

**Table 3-412. Function fac\_float\_preload**

Function name	fac_fixed_data_preload
Function prototype	void fac_fixed_data_preload(uint8_t size, int16_t *data);
Function descriptions	FAC preload fixed data pointer
Precondition	-
The called functions	-

Input parameter{in}	
size	size of data
Input parameter{in}	
*data	16-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC preload context of the coefficient vector B */
```

```
fac_fixed_data_preload_struct init_struct;
```

```
fac_fixed_data_preload(init_struct->coeffB_size,(init_struct->coeffB_ctx));
```

### fac\_float\_data\_preload

The description of fac\_float\_data\_preload is shown as below:

**Table 3-413. Function fac\_float\_preload**

Function name	fac_float_data_preload
Function prototype	void fac_float_data_preload(uint8_t size, float *data);
Function descriptions	FAC preload float data pointer
Precondition	-
The called functions	-
Input parameter{in}	
size	size of data
Input parameter{in}	
*data	32-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC preload context of the coefficient vector B */
```

```
fac_float_data_preload_struct init_struct;
```

```
fac_float_data_preload (init_struct->coeffB_size,(init_struct->coeffB_ctx));
```

### fac\_reset

The description of fac\_reset is shown as below:



Table 3-414. Function `fac_init`

Function name	<code>fac_reset</code>
Function prototype	<code>void fac_reset(void);</code>
Function descriptions	FAC reset
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC reset write and read pointers */
```

```
fac_reset();
```

### `fac_clip_config`

The description of `fac_clip_config` is shown as below:

Table 3-415. Function `fac_clip_config`

Function name	<code>fac_clip_config</code>
Function prototype	<code>void fac_clip_config(uint8_t cpmode);</code>
Function descriptions	config the FAC clip feature
Precondition	-
The called functions	-
Input parameter{in}	
<code>cpmode</code>	the state of clip
<code>FAC_CP_ENABLED</code>	enable clip
<code>FAC_CP_DISABLE</code>	disable clip
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config the FAC clip enable */
```

```
fac_clip_config(FAC_CP_ENABLE);
```

### `fac_float_enable`

The description of `fac_float_enable` is shown as below:

Table 3-416. Function fac\_init

Function name	fac_float_enable
Function prototype	void fac_float_enable(void);
Function descriptions	enable FAC float point format
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable FAC float point format */
fac_float_enable ();
```

### fac\_float\_disable

The description of fac\_float\_disable is shown as below:

Table 3-417. Function fac\_float\_disable

Function name	fac_float_disable
Function prototype	void fac_float_disable(void);
Function descriptions	disable FAC float point format
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable FAC float point format */
fac_float_disable ();
```

### fac\_dma\_enable

The description of fac\_dma\_enable is shown as below:

Table 3-418. Function fac\_dma\_enable

Function name	fac_dma_enable
Function prototype	void fac_dma_enable(uint32_t dma_req);
Function descriptions	enable the FAC DMA
Precondition	-
The called functions	-
Input parameter{in}	
dma_req	transfer type
FAC_DMA_READ	enable DMA read buffer
FAC_DMA_WRITE	enable DMA write buffer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the FAC read buffer by DMA */
```

```
fac_dma_enable(FAC_DMA_READ);
```

### fac\_dma\_disable

The description of fac\_dma\_disable is shown as below:

Table 3-419. Function fac\_dma\_disable

Function name	fac_dma_disable
Function prototype	void fac_dma_disable(uint32_t dma_req);
Function descriptions	disable the FAC DMA
Precondition	-
The called functions	-
Input parameter{in}	
dma_req	transfer type
FAC_DMA_READ	disable DMA read buffer
FAC_DMA_WRITE	disable DMA write buffer
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the FAC read buffer by DMA */
```

```
fac_dma_disable(FAC_DMA_READ);
```

## fac\_x0\_config

The description of fac\_x0\_config is shown as below:

**Table 3-420. Function fac\_dma\_enable**

<b>Function name</b>	fac_x0_config
<b>Function prototype</b>	void fac_x0_config(uint32_t watermark, uint8_t baseaddr, uint8_t bufsize);
<b>Function descriptions</b>	FAC config input buffer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>watermark</b>	threshold of input buffer
<i>FAC_THRESHOLD_x</i>	X0 buffer full threshold(x =1,2,4,8)
<b>Input parameter{in}</b>	
<b>baseaddr</b>	base address of input buffer, 0..255
<b>Input parameter{in}</b>	
<b>bufsize</b>	buffer size of input buffer, 0..255
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config input buffer threshold of watermark, base address, buffer size */
```

```
fac_x0_config(FAC_THRESHOLD_1,20,10);
```

## fac\_x1\_config

The description of fac\_x1\_config is shown as below:

**Table 3-421. Function fac\_dma\_enable**

<b>Function name</b>	fac_x1_config
<b>Function prototype</b>	void fac_x1_config(uint8_t baseaddr, uint8_t bufsize);
<b>Function descriptions</b>	FAC config coefficient buffer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>baseaddr</b>	base address of coefficientbuffer, 0..255
<b>Input parameter{in}</b>	
<b>bufsize</b>	buffer size of coefficient buffer, 0..255
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config coefficient buffer base address and buffer size */
```

```
fac_x1_config(10,10);
```

### fac\_y\_config

The description of fac\_y\_config is shown as below:

**Table 3-422. Function fac\_dma\_enable**

<b>Function name</b>	fac_y_config
<b>Function prototype</b>	void fac_y_config(uint32_t watermark, uint8_t baseaddr, uint8_t bufsize);
<b>Function descriptions</b>	FAC config output buffer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>watermark</b>	threshold of output buffer
<i>FAC_THRESHOLD_x</i>	Y buffer empty threshold(x =1,2,4,8)
<b>Input parameter{in}</b>	
<b>baseaddr</b>	base address of outputbuffer, 0..255
<b>Input parameter{in}</b>	
<b>bufsize</b>	buffer size of output buffer, 0..255
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config output buffer threshold of watermark, base address, buffer size */
```

```
fac_y_config(FAC_THRESHOLD_1,30,20);
```

### fac\_function\_config

The description of fac\_function\_config is shown as below:

**Table 3-423. Function fac\_dma\_enable**

<b>Function name</b>	fac_function_config
<b>Function prototype</b>	void fac_function_config(fac_parameter_struct* fac_parameter);
<b>Function descriptions</b>	FAC config function execute
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fac_parameter</b>	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_parameter_struct</a> .

Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* config FAC work in FIR mode*/

fac_parameter_struct facconfig;

facconfig.func = FUNC_CONVO_FIR;

facconfig.ipp = fir_coeffb_size;

facconfig.ipq = 0;

facconfig.ipr = fir_gain;

fac_function_config(&facconfig);

```

## fac\_start

The description of fac\_start is shown as below:

**Table 3-424. Function fac\_start**

Function name	fac_start
Function prototype	void fac_start(void);
Function descriptions	start the FAC
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* start the FAC*/

fac_start();

```

## fac\_stop

The description of fac\_stop is shown as below:

**Table 3-425. Function fac\_start**

Function name	fac_stop
---------------	----------

<b>Function prototype</b>	void fac_stop(void);
<b>Function descriptions</b>	stop the FAC
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* stop the FAC*/
```

```
fac_stop();
```

### fac\_finish\_calculate

The description of fac\_finish\_calculate is shown as below:

**Table 3-426. Function fac\_start**

<b>Function name</b>	fac_finish_calculate
<b>Function prototype</b>	void fac_finish_calculate(void);
<b>Function descriptions</b>	finish the filter calculate
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* finish the filter calculate*/
```

```
fac_finish_calculate ();
```

### fac\_fixed\_data\_write

The description of fac\_fixed\_data\_write is shown as below:

**Table 3-427. Function fac\_fixed\_data\_write**

<b>Function name</b>	fac_fixed_data_write
<b>Function prototype</b>	void fac_fixed_data_write(int16_t data);

<b>Function descriptions</b>	FAC write data with fixed point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data</b>	16-bit data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* FAC fixed data write */
fac_fixed_data_write(300);
```

### fac\_fixed\_data\_read

The description of fac\_float\_data\_read is shown as below:

**Table 3-428. Function fac\_fixed\_data\_write**

<b>Function name</b>	fac_fixed_data_read
<b>Function prototype</b>	int16_t fac_fixed_data_read(void);
<b>Function descriptions</b>	FAC read data with fixed point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int16_t</b>	FAC int16_t data value

Example:

```
/* FAC fixed data read */
int16_t data;
data = fac_fixed_data_read();
```

### fac\_float\_data\_write

The description of fac\_float\_data\_write is shown as below:

**Table 3-429. Function fac\_float\_data\_write**

<b>Function name</b>	fac_float_data_write
<b>Function prototype</b>	void fac_float_data_write(float data);



<b>Function descriptions</b>	FAC write data with float ponit format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data</b>	32-bit data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* FAC float data write */
fac_float_data_write(200.0f);
```

### fac\_float\_data\_read

The description of fac\_float\_data\_read is shown as below:

**Table 3-430. Function fac\_fixed\_data\_write**

<b>Function name</b>	fac_float_data_read
<b>Function prototype</b>	int16_t fac_float_data_read(void);
<b>Function descriptions</b>	FAC read data with float point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>float</b>	FAC float data value

Example:

```
/* FAC float data read */
float data;
data = fac_float_data_read();
```

### fac\_interrupt\_enable

The description of fac\_interrupt\_enable is shown as below:

**Table 3-431. Function fac\_interrupt\_enable**

<b>Function name</b>	fac_interrupt_enable
<b>Function prototype</b>	void fac_interrupt_enable(uint32_t interrupt);

<b>Function descriptions</b>	enable the FAC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FAC interrupt
<i>FAC_CTL_RIE</i>	read buffer interrupt
<i>FAC_CTL_WIE</i>	write buffer interrupt
<i>FAC_CTL_OFEIE</i>	overflow error interrupt
<i>FAC_CTL_UFEIE</i>	underflow error interrupt
<i>FAC_CTL_STEIE</i>	saturation error interrupt
<i>FAC_CTL_GSTEIE</i>	gain saturation error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the FAC read buffer interrupt*/
```

```
fac_interrupt_enable(FAC_CTL_RIE);
```

### fac\_interrupt\_disable

The description of fac\_interrupt\_disable is shown as below:

**Table 3-432. Function fac\_interrupt\_disable**

<b>Function name</b>	fac_interrupt_disable
<b>Function prototype</b>	void fac_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable the FAC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FAC interrupt
<i>FAC_CTL_RIE</i>	read buffer interrupt
<i>FAC_CTL_WIE</i>	write buffer interrupt
<i>FAC_CTL_OFEIE</i>	overflow error interrupt
<i>FAC_CTL_UFEIE</i>	underflow error interrupt
<i>FAC_CTL_STEIE</i>	saturation error interrupt
<i>FAC_CTL_GSTEIE</i>	gain saturation error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the FAC read buffer interrupt*/
```

```
fac_interrupt_disable(FAC_CTL_RIE);
```

### fac\_interrupt\_flag\_get

The description of fac\_interrupt\_flag\_get is shown as below:

**Table 3-433. Function fac\_interrupt\_flag\_get**

<b>Function name</b>	fac_interrupt_flag_get
<b>Function prototype</b>	FlagStatus fac_interrupt_flag_get(uint8_t interrupt);
<b>Function descriptions</b>	get FAC interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FAC interrupt flag status
FAC_INT_FLAG_YBEF	Y buffer empty interrupt flag
FAC_INT_FLAG_X0BF F	X0 buffer full interrupt flag
FAC_INT_FLAG_OFEF	overflow error interrupt flag
FAC_INT_FLAG_UFEF	underflow error interrupt flag
FAC_INT_FLAG_STEF	saturation error interrupt flag
FAC_INT_FLAG_GSTE F	gain saturation error interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get Y buffer empty flag status */
```

```
fac_interrupt_flag_get(FAC_INT_FLAG_YBEF);
```

### fac\_flag\_get

The description of fac\_flag\_get is shown as below:

**Table 3-434. Function fac\_flag\_get**

<b>Function name</b>	fac_flag_get
<b>Function prototype</b>	FlagStatus fac_flag_get(uint32_t flag);
<b>Function descriptions</b>	get FAC flag status
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>interrupt</b>	FAC interrupt flag status
<i>FAC_FLAG_YBEF</i>	Y buffer empty flag
<i>FAC_FLAG_X0BFF</i>	X0 buffer full flag
<i>FAC_FLAG_OFEF</i>	overflow error flag
<i>FAC_FLAG_UFEF</i>	underflow error flag
<i>FAC_FLAG_STEF</i>	saturation error flag
<i>FAC_FLAG_GSTEF</i>	gain saturation error flag
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get Y buffer empty flag status */
fac_flag_get(FAC_FLAG_YBEF);
```

## 3.15. FFT

The Fast Fourier Transform (FFT) is an efficient computation of the Discrete Fourier Transform (DFT). The module supports CPU to offload FFT operations. Compared to a software implementation, it can accelerate calculations and time critical tasks. The module supports 6 configuration FFT point number up to 1024, and input and output data should be IEEE-754 single precision float point complex number. The FFT registers are listed in chapter [3.15.1](#), the FFT firmware functions are introduced in chapter [3.15.2](#).

### 3.15.1. Descriptions of Peripheral registers

FFT registers are listed in the table shown as below:

**Table 3-435. FFT Registers**

Registers	Descriptions
FFT_CSRT	Control and status register
FFT_RESADDR	Real start address register
FFT_IMSADDR	Image start address register
FFT_WSADDR	Window start address register
FFT_OSADDR	Output start address register
FFT_LOOPLEN	Loop length register

### 3.15.2. Descriptions of Peripheral functions

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

Table 3-436. FFT firmware function

Function name	Function description
fft_deinit	reset the FFT peripheral
fft_struct_para_init	initialize the parameters of FFT struct with the default values
fft_init	initialize the FFT peripheral for calculation
fft_calculation_start	start FFT calculation
fft_calculation_stop	stop FFT calculation
fft_point_number_config	configure FFT calculation point number
fft_mode_config	configure FFT mode
fft_window_enable	enable FFT window function
fft_window_disable	disable FFT window function
fft_downsample_config	configure FFT down sample value
fft_image_source_config	configure FFT image part source
fft_real_addr_config	configure real part start address
fft_image_addr_config	configure image part start address
fft_window_addr_config	configure window function start address
fft_output_addr_config	configure output start address
fft_loop_buffer_length_config	configure loop buffer length
fft_loop_buffer_index_config	configure loop buffer index
fft_flag_get	get FFT flag status
fft_flag_clear	clear FFT flag bit
fft_interrupt_enable	enable FFT interrupt
fft_interrupt_disable	disable FFT interrupt
fft_interrupt_flag_get	get FFT interrupt flag status
fft_interrupt_flag_clear	clear FFT interrupt flag bit

### Structure fft\_parameter\_struct

Table 3-437. Structure fft\_parameter\_struct

Member name	Function description
mode_sel	FFT mode configure (FFT_MODE, IFFT_MODE)
point_num	FFT point number select (FFT_POINT_x, x = 32, 64, 128, 256, 512, 1024)
window_enable	FFT window function enable (FFT_WINDOW_ENABLE, FFT_WINDOW_DISABLE)
downsamp_sel	FFT input data down sample selection (FFT_DOWNSAMPLE_x, x = 1..16)
image_source	FFT image input source select (FFT_IM_MEMORY, FFT_IM_ZERO, FFT_IM_MEMORY_OPPOSITE)
loopbuf_len	FFT loop buffer length
loopbuf_index	FFT loop buffer index
real_addr	FFT real start address
image_addr	FFT image start address
window_addr	FFT window start address
output_addr	FFT output start address

## fft\_deinit

The description of fft\_deinit is shown as below:

**Table 3-438. Function fft\_deinit**

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

Example:

```
/* reset FFT */
fft_deinit();
```

## fft\_struct\_para\_init

The description of fft\_struct\_para\_init is shown as below:

**Table 3-439. Function fft\_struct\_para\_init**

<b>Function name</b>	fft_struct_para_init
<b>Function prototype</b>	void fft_struct_para_init(fft_parameter_struct* fft_parameter);
<b>Function descriptions</b>	initialize the parameters of FFT struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
fft_parameter	FFT init parameter struct pointer, the structure members can refer to <a href="#">fft_struct_para_init</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the FFT filter parameter struct */
fft_parameter_struct fftconfig;
fft_struct_para_init(&fftconfig);
```

## fft\_init

The description of fft\_init is shown as below:

**Table 3-440. Function fft\_init**

<b>Function name</b>	fft_init
<b>Function prototype</b>	void fft_init(fft_parameter_struct* fft_parameter);
<b>Function descriptions</b>	initialize the FFT peripheral for calculation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fft_parameter</b>	FFT init parameter struct pointer, the structure members can refer to <a href="#">fft_struct_para_init</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* config FFT parameter */

fft_parameter_struct fftconfig;

float fft_real_buf[32] = {0};

float fft_output_buf[32] = {0};

fft_struct_para_init(&fftconfig);

fftconfig.mode_sel      = FFT_MODE;

fftconfig.point_num     = FFT_POINT_1024;

fftconfig.window_enable = FFT_WINDOW_DISABLE;

fftconfig.downsamp_sel  = FFT_DOWNSAMPLE_8;

fftconfig.image_source  = FFT_IM_ZERO;

fftconfig.loopbuf_len   = 32U;

fftconfig.loopbuf_index = 0U;

fftconfig.real_addr     = (uint32_t)fft_real_buf;

fftconfig.image_addr    = NULL;

fftconfig.window_addr   = NULL;

fftconfig.output_addr   = (uint32_t)fft_output_buf;

fft_init(&fftconfig);

```

## fft\_calculation\_start

The description of fft\_calculation\_start is shown as below:

**Table 3-441. Function fft\_calculation\_start**

<b>Function name</b>	fft_calculation_start
<b>Function prototype</b>	void fft_calculation_start(void);
<b>Function descriptions</b>	start FFT calculation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* start FFT calculation */
fft_calculation_start();
```

## fft\_calculation\_stop

The description of fft\_calculation\_stop is shown as below:

**Table 3-442. Function fft\_calculation\_stop**

<b>Function name</b>	fft_calculation_stop
<b>Function prototype</b>	void fft_calculation_stop(void);
<b>Function descriptions</b>	stop FFT calculation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* stop FFT calculation */
fft_calculation_stop();
```



## fft\_point\_number\_config

The description of `fft_point_number_config` is shown as below:

**Table 3-443. Function `fft_point_number_config`**

<b>Function name</b>	<code>fft_point_number_config</code>
<b>Function prototype</b>	<code>void fft_point_number_config(uint8_t point_num);</code>
<b>Function descriptions</b>	configure FFT calculation point number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>point_num</b>	point number
<i>FFT_POINT_32</i>	FFT point number is 32
<i>FFT_POINT_64</i>	FFT point number is 64
<i>FFT_POINT_128</i>	FFT point number is 128
<i>FFT_POINT_256</i>	FFT point number is 256
<i>FFT_POINT_512</i>	FFT point number is 512
<i>FFT_POINT_1024</i>	FFT point number is 1024
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure FFT calculation point number */
fft_point_number_config (FFT_POINT_1024);
```

## fft\_mode\_config

The description of `fft_mode_config` is shown as below:

**Table 3-444. Function `fft_mode_config`**

<b>Function name</b>	<code>fft_mode_config</code>
<b>Function prototype</b>	<code>void fft_mode_config(uint32_t mode);</code>
<b>Function descriptions</b>	configure FFT mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mode</b>	calculation mode
<i>FFT_MODE</i>	excute FFT operation
<i>IFFT_MODE</i>	excute FFT inverse operation
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* excute FFT operation mode */
```

```
fft_mode_config (FFT_MODE);
```

### fft\_window\_enable

The description of fft\_window\_enable is shown as below:

**Table 3-445. Function fft\_window\_enable**

<b>Function name</b>	fft_window_enable
<b>Function prototype</b>	void fft_window_enable(void);
<b>Function descriptions</b>	enable FFT window function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FFT window function */
```

```
fft_window_enable();
```

### fft\_window\_disable

The description of fft\_window\_disable is shown as below:

**Table 3-446. Function fft\_window\_disable**

<b>Function name</b>	fft_window_disable
<b>Function prototype</b>	void fft_window_disable(void);
<b>Function descriptions</b>	disable FFT window function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable FFT window function */
```

```
fft_window_disable();
```

### fft\_downsample\_config

The description of fft\_downsample\_config is shown as below:

**Table 3-447. Function fft\_downsample\_config**

<b>Function name</b>	fft_downsample_config
<b>Function prototype</b>	void fft_downsample_config(uint8_t sample_value);
<b>Function descriptions</b>	configure FFT down sample value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sample_value</b>	down sample value
<i>FFT_DOWNSAMPLE_x</i> (x=1..16)	down sample value, x=1..16
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure FFT down sample value */
```

```
fft_downsample_config (FFT_DOWNSAMPLE_16);
```

### fft\_image\_source\_config

The description of fft\_image\_source\_config is shown as below:

**Table 3-448. Function fft\_image\_source\_config**

<b>Function name</b>	fft_image_source_config
<b>Function prototype</b>	void fft_image_source_config(uint8_t im_src);
<b>Function descriptions</b>	configure FFT image part source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>im_src</b>	image source type
<i>FFT_IM_MEMORY</i>	image input is from FFT_IMSADDR
<i>FFT_IM_ZERO</i>	image input equal to zero

<i>FFT_IM_MEMORY_OP</i> <i>POSITE</i>	image input is the opposite number of image data from FFT_IMSADDR
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure FFT image part source */
```

```
fft_image_source_config (FFT_IM_MEMORY);
```

### fft\_real\_addr\_config

The description of `fft_real_addr_config` is shown as below:

**Table 3-449. Function `fft_real_addr_config`**

<b>Function name</b>	<code>fft_real_addr_config</code>
<b>Function prototype</b>	<code>void fft_real_addr_config(uint32_t addr);</code>
<b>Function descriptions</b>	configure real part start address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>addr</b>	real memory address
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure real part start address */
```

```
float fft_real_buf[32] = {0};
```

```
fft_real_addr_config ((uint32_t)fft_real_buf);
```

### fft\_image\_addr\_config

The description of `fft_image_addr_config` is shown as below:

**Table 3-450. Function `fft_image_addr_config`**

<b>Function name</b>	<code>fft_image_addr_config</code>
<b>Function prototype</b>	<code>void fft_image_addr_config(uint32_t addr);</code>
<b>Function descriptions</b>	configure image part start address
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>addr</b>	image memory address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure image part start address */

float fft_image_buf[32] = {0};

fft_image_addr_config ((uint32_t)fft_image_buf);
```

### fft\_window\_addr\_config

The description of fft\_window\_addr\_config is shown as below:

**Table 3-451. Function fft\_window\_addr\_config**

<b>Function name</b>	fft_window_addr_config
<b>Function prototype</b>	void fft_window_addr_config(uint32_t addr);
<b>Function descriptions</b>	configure window start address
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>addr</b>	window memory address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure window part start address */

float fft_window_buf[32] = {0};

fft_window_addr_config ((uint32_t)fft_window_buf);
```

### fft\_output\_addr\_config

The description of fft\_output\_addr\_config is shown as below:

**Table 3-452. Function fft\_output\_addr\_config**

<b>Function name</b>	fft_output_addr_config
<b>Function prototype</b>	void fft_output_addr_config(uint32_t addr);
<b>Function descriptions</b>	configure output start address
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
addr	output memory address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure output start address */
```

```
float fft_output_buf[32] = {0};
```

```
fft_output_addr_config ((uint32_t)fft_output_buf);
```

### fft\_loop\_buffer\_length\_config

The description of fft\_loop\_buffer\_length\_config is shown as below:

**Table 3-453. Function fft\_loop\_buffer\_length\_config**

Function name	fft_loop_buffer_length_config
Function prototype	void fft_loop_buffer_length_config(uint16_t length);
Function descriptions	configure loop buffer length
Precondition	-
The called functions	-
Input parameter{in}	
length	loop buffer length
0x0000~0xFFFF	loop buffer length value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure loop buffer length */
```

```
fft_loop_buffer_length_config (10U);
```

### fft\_loop\_buffer\_index\_config

The description of fft\_loop\_buffer\_index\_config is shown as below:

**Table 3-454. Function fft\_loop\_buffer\_index\_config**

Function name	fft_loop_buffer_index_config
Function prototype	void fft_loop_buffer_index_config(uint16_t index);
Function descriptions	configure loop buffer index

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>index</b>	index value
0x0000~LENGTH[15:0]	it can not more than LENGTH[15:0].
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure loop buffer index value*/
```

```
fft_loop_buffer_index_config (0U);
```

### fft\_flag\_get

The description of fft\_flag\_get is shown as below:

**Table 3-455. Function fft\_flag\_get**

<b>Function name</b>	fft_flag_get
<b>Function prototype</b>	FlagStatus fft_flag_get(uint32_t flag);
<b>Function descriptions</b>	get FFT flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	the flag bits
FFT_FLAG_DMABSY	DMA busy flag
FFT_FLAG_CCF	FFT calculation completion flag
FFT_FLAG_TAEIF	FFT transfer access error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get FFT flag status */
```

```
FlagStatus flag_value;
```

```
flag_value = fft_flag_get(FFT_FLAG_DMABSY);
```

### fft\_flag\_clear

The description of fft\_flag\_clear is shown as below:

Table 3-456. Function `fft_flag_clear`

Function name	<code>fft_flag_clear</code>
Function prototype	<code>void fft_flag_clear(uint32_t flag);</code>
Function descriptions	clear FFT flag bit
Precondition	-
The called functions	-
Input parameter{in}	
flag	the flag bits
<code>FFT_FLAG_CCF</code>	FFT calculation completion flag
<code>FFT_FLAG_TAEIF</code>	FFT transfer access error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear FFT flag status */
fft_flag_clear (FFT_FLAG_CCF);
```

### `fft_interrupt_enable`

The description of `fft_interrupt_enable` is shown as below:

Table 3-457. Function `fft_interrupt_enable`

Function name	<code>fft_interrupt_enable</code>
Function prototype	<code>void fft_interrupt_enable(uint32_t fft_interrupt);</code>
Function descriptions	enable FFT interrupt
Precondition	-
The called functions	-
Input parameter{in}	
fft_interrupt	the FFT interrupt
<code>FFT_INT_CCF</code>	FFT calculation completion interrupt
<code>FFT_INT_TAEIF</code>	FFT transfer access error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable FFT calculation completion interrupt */
fft_interrupt_enable (FFT_INT_CCF);
```



## fft\_interrupt\_disable

The description of `fft_interrupt_disable` is shown as below:

**Table 3-458. Function `fft_interrupt_disable`**

<b>Function name</b>	<code>fft_interrupt_disable</code>
<b>Function prototype</b>	<code>void fft_interrupt_disable (uint32_t fft_interrupt);</code>
<b>Function descriptions</b>	disable FFT interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fft_interrupt</b>	the FFT interrupt
<i>FFT_INT_CCF</i>	FFT calculation completion interrupt
<i>FFT_INT_TAEIF</i>	FFT transfer access error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FFT calculation completion interrupt */
```

```
fft_interrupt_disable (FFT_INT_CCF);
```

## fft\_interrupt\_flag\_get

The description of `fft_interrupt_flag_get` is shown as below:

**Table 3-459. Function `fft_interrupt_flag_get`**

<b>Function name</b>	<code>fft_interrupt_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus fft_interrupt_flag_get(uint32_t int_flag);</code>
<b>Function descriptions</b>	get FFT interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	FFT interrupt flag
<i>FFT_INT_FLAG_CCF</i>	FFT calculation completion interrupt flag
<i>FFT_INT_FLAG_TAEIF</i>	FFT transfer access error interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get FFT interrupt flag status */
```

FlagStatus flag\_value;

```
flag_value = fft_interrupt_flag_get(FFT_INT_FLAG_CCF);
```

### fft\_interrupt\_flag\_clear

The description of fft\_interrupt\_flag\_clear is shown as below:

**Table 3-460. Function fft\_interrupt\_flag\_clear**

<b>Function name</b>	fft_interrupt_flag_clear
<b>Function prototype</b>	void fft_interrupt_flag_clear(uint32_t int_flag);
<b>Function descriptions</b>	clear FFT interrupt flag bit
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	FFT interrupt flag
<i>FFT_INT_FLAG_CCF</i>	FFT calculation completion interrupt flag
<i>FFT_INT_FLAG_TAEIF</i>	FFT transfer access error interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear FFT interrupt flag bit */
```

```
fft_interrupt_flag_clear(FFT_INT_FLAG_CCF);
```

## 3.16. FMC

There is flash controller and option byte. The FMC registers are listed in chapter [3.16.1](#) the FMC firmware functions are introduced in chapter [3.16.2](#).

### 3.16.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

**Table 3-461. FMC Registers**

Registers	Descriptions
FMC_WS	FMC wait state register
FMC_RUNKEY	FMC unlock flash mode during run mode key register
FMC_KEY	FMC unlock key register
FMC_OBKEY	FMC option bytes unlock key register
FMC_STAT	FMC status register
FMC_CTL	FMC control register

Registers	Descriptions
FMC_ECCCS	FMC ECC control and status register
FMC_OBCTL	FMC FMC option byte control register
FMC_DCRP_SADD R0	FMC DCRP area start address register 0
FMC_DCRP_EADD R0	FMC DCRP area end address register 0
FMC_BK0WP0	FMC bank0 erase/program protection area 0 register
FMC_BK0WP1	FMC bank0 erase/program protection area 1 register
FMC_DCRP_SADD R1	FMC DCRP area start address register 1
FMC_DCRP_EADD R1	FMC DCRP area end address register 1
FMC_BK1WP0	FMC bank1 erase/program protection area 0 register
FMC_BK1WP1	FMC bank1 erase/program protection area 1 register
FMC_BK0SCR	FMC bank0 securable area register
FMC_BK1SCR	FMC bank1 securable area register
FMC_PID	FMC product ID register

### 3.16.2. Descriptions of Peripheral functions

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

**Table 3-462. FMC firmware function**

Function name	Function description
fmc_unlock	unlock the main FMC operation
fmc_lock	lock the main FMC operation
fmc_wscnt_set	set the wait state
fmc_prefetch_enable	enable pre-fetch
fmc_prefetch_disable	disable pre-fetch
fmc_icache_enable	enable IBUS cache
fmc_icache_disable	disable IBUS cache
fmc_icache_reset	reset IBUS cache
fmc_dcache_enable	enable DBUS cache
fmc_dcache_disable	disable DBUS cache
fmc_dcache_reset	reset DBUS cache
fmc_page_erase	erase page
fmc_bank0_erase	erase bank0
fmc_bank1_erase	erase bank1
fmc_mass_erase	erase whole chip
fmc_doubleword_program	program a doubleword at the given address
fmc_bank0_base_address_get	get base address of bank0
fmc_bank1_base_address_get	get base address of bank1

Function name	Function description
fmc_page_size_get	get page size
fmc_debugger_enable	enable debugger
fmc_debugger_disable	disable debugger
fmc_slp_pd_mode_enable	flash enter power down mode when MCU enter sleep mode
fmc_slp_pd_mode_disable	flash exit power down mode when MCU enter sleep mode
fmc_scr_area_enable	enable secure area protection
ob_unlock	unlock the option bytes operation
ob_lock	lock the option bytes operation
ob_reload	reload the option bytes operation
ob_user_write	program option bytes USER
ob_security_protection_config	configure the option byte security protection
ob_dcrp_config	configure the option bytes DCRP area
ob_write_protection_config	configure the option bytes write protection area
ob_scr_area_config	configure the option bytes secure area
ob_boot_lock_config	configure the option byte boot lock
ob_bank_memory_swap_config	configure FMC memory mapping swap bit
ob_user_get	get the value of option bytes USER
ob_bor_threshold_get	get option byte BOR threshold value
ob_security_protection_level_get	get the value of option bytes security protection level in FMC_OBCTL register
ob_dcrp_area_get	get DCRP area configuration
ob_write_protection_get	get the value of option bytes write protection
ob_scr_area_size_get	get size of secure area
ob_boot_config_get	get boot configuration
fmc_ecc_flag_get	get ECC flag status
fmc_ecc_flag_clear	clear ECC flag status
fmc_eccor_interrupt_enable	enable ECCOR interrupt
fmc_eccor_interrupt_disable	disable ECCOR interrupt
fmc_eccor_interrupt_flag_get	get ECCOR interrupt flag
fmc_eccor_interrupt_flag_clear	clear ECCOR interrupt flag
fmc_flag_get	get FMC flag status
fmc_flag_clear	clear the FMC flag
fmc_interrupt_enable	enable FMC interrupt
fmc_interrupt_disable	disable FMC interrupt
fmc_interrupt_flag_get	get FMC interrupt flag
fmc_interrupt_flag_clear	clear FMC interrupt flag
fmc_state_get	return FMC state
fmc_ready_wait	check FMC ready or not
fmc_pd_mode_enter	flash enter power down mode when MCU run mode
fmc_pd_mode_exit	flash exit power down mode when MCU run mode
ob_bank_mode_config	configure the option byte bank mode

## Enum fmc\_state\_enum

**Table 3-463. Enum fmc\_state\_enum**

enum name	enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_OBERR	option byte read error
FMC_RPERR	read protection error
FMC_PGSERR	program sequence error
FMC_PGMERR	program size error
FMC_PGAERR	program alignment error
FMC_WPERR	erase/program protection error
FMC_PGERR	program error
FMC_OPRERR	operation error
FMC_TOERR	timeout error
FMC_OB_HSPC	high security protection
FMC_OB_LSPC	low security protection

## fmc\_unlock

The description of fmc\_unlock is shown as below:

**Table 3-464. Function fmc\_unlock**

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

Example:

```
/* unlock FMC_CTL register */
fmc_unlock();
```

## fmc\_lock

The description of fmc\_lock is shown as below:

Table 3-465. Function `fmc_lock`

Function name	<code>fmc_lock</code>
Function prototype	<code>void fmc_lock(void);</code>
Function descriptions	lock FMC_CTL register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock FMC_CTL register */
```

```
fmc_lock();
```

### `fmc_wscnt_set`

The description of `fmc_wscnt_set` is shown as below:

Table 3-466. Function `fmc_wscnt_set`

Function name	<code>fmc_wscnt_set</code>
Function prototype	<code>void fmc_wscnt_set(uint32_t wscnt);</code>
Function descriptions	set the wait state
Precondition	-
The called functions	-
Input parameter{in}	
<b>wscnt</b>	wait state
<code>FMC_WAIT_STATE_0</code>	0 wait state added
<code>FMC_WAIT_STATE_1</code>	1 wait state added
<code>FMC_WAIT_STATE_2</code>	2 wait state added
<code>FMC_WAIT_STATE_3</code>	3 wait state added
<code>FMC_WAIT_STATE_4</code>	4 wait state added
<code>FMC_WAIT_STATE_5</code>	5 wait state added
<code>FMC_WAIT_STATE_6</code>	6 wait state added
<code>FMC_WAIT_STATE_7</code>	7 wait state added
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the wait state 0 */
```

```
fmc_wsnt_set(FMC_WAIT_STATE_0);
```

### fmc\_prefetch\_enable

The description of fmc\_prefetch\_enable is shown as below::

**Table 3-467. Function fmc\_prefetch\_enable**

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

Example:

```
/* enable pre-fetch */
```

```
fmc_prefetch_enable( );
```

### fmc\_prefetch\_disable

The description of fmc\_prefetch\_disable is shown as below::

**Table 3-468. Function fmc\_prefetch\_disable**

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

Example:

```
/* disable pre-fetch */
```

```
fmc_prefetch_disable( );
```

### fmc\_icache\_enable

The description of fmc\_icache\_enable is shown as below::

**Table 3-469. Function fmc\_icache\_enable**

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

Example:

```
/* enable icache */
fmc_icache_enable( );
```

### fmc\_icache\_disable

The description of fmc\_icache\_disable is shown as below::

**Table 3-470. Function fmc\_icache\_disable**

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

Example:

```
/* disable icache */
fmc_icache_disable( );
```



## fmc\_icache\_reset

The description of fmc\_icache\_reset\_enable is shown as below::

**Table 3-471. Function fmc\_icache\_reset\_enable**

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

Example:

```
/* icache reset */
fmc_icache_reset( );
```

## fmc\_dcache\_enable

The description of fmc\_dcache\_enable is shown as below::

**Table 3-472. Function fmc\_dcache\_enable**

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

Example:

```
/* enable dcache */
fmc_dcache_enable( );
```

## fmc\_dcache\_disable

The description of fmc\_dcache\_disable is shown as below::

**Table 3-473. Function fmc\_dcache\_disable**

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

Example:

```
/* disable dcache */
fmc_dcache_disable( );
```

## fmc\_dcache\_reset

The description of fmc\_dcache\_reset\_enable is shown as below::

**Table 3-474. Function fmc\_dcache\_reset\_enable**

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

Example:

```
/* dcache reset */
fmc_dcache_reset( );
```

## fmc\_page\_erase

The description of fmc\_page\_erase is shown as below:

**Table 3-475. Function fmc\_page\_erase**

<b>Function name</b>	fmc_page_erase
<b>Function prototype</b>	fmc_page_erase(uint32_t bank, uint32_t page_number_in_bank)
<b>Function descriptions</b>	FMC erase page
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>bank</b>	specify which bank the target page is in
<i>FMC_BANK0</i>	BANK0 of main flash
<i>FMC_BANK1</i>	BANK1 of main flash
<b>Input parameter{in}</b>	
<b>page_number_in_bank</b>	Page number in the bank
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();

/* erase page */

fmc_state_enum state = fmc_page_erase(FMC_BANK0, 0);
```

## fmc\_bank0\_erase

The description of fmc\_bank0\_erase is shown as below:

**Table 3-476. Function fmc\_bank0\_erase**

<b>Function name</b>	fmc_bank0_erase
<b>Function prototype</b>	fmc_state_enum fmc_bank0_erase(void);
<b>Function descriptions</b>	FMC erase bank0. It is valid only in dual bank mode.
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();

/* erase bank0 */

fmc_state_enum state = fmc_bank0_erase( );
```

### fmc\_bank1\_erase

The description of fmc\_bank1\_erase is shown as below:

**Table 3-477. Function fmc\_bank1\_erase**

<b>Function name</b>	fmc_bank1_erase
<b>Function prototype</b>	fmc_state_enum fmc_bank1_erase(void);
<b>Function descriptions</b>	FMC erase bank1. It is valid only in dual bank mode.
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
fmc_state_enum	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();

/* erase bank1 */

fmc_state_enum state = fmc_bank1_erase( );
```

### fmc\_mass\_erase

The description of fmc\_mass\_erase is shown as below:

**Table 3-478. Function fmc\_mass\_erase**

<b>Function name</b>	fmc_mass_erase
<b>Function prototype</b>	fmc_state_enum fmc_mass_erase(void);
<b>Function descriptions</b>	FMC erase whole chip
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>
-----------------------	--

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 3-479. Function fmc\_doubleword\_program**

<b>Function name</b>	fmc_doubleword_program
<b>Function prototype</b>	fmc_state_enum fmc_doubleword_program(uint32_t address, uint64_t data);
<b>Function descriptions</b>	program a double word at the given address in main flash
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>address</b>	address to program
<b>Input parameter{in}</b>	
<b>data</b>	double word to program
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

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\_bank0\_base\_address\_get

The description of fmc\_bank0\_base\_address\_get is shown as below:

**Table 3-480. Function fmc\_bank0\_base\_address\_get**

<b>Function name</b>	fmc_bank0_base_address_get
<b>Function prototype</b>	uint32_t fmc_bank0_base_address_get(void);
<b>Function descriptions</b>	get base address of bank0

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	base address of bank0

Example:

```
/* get base address of bank0 */
uint32_t base_addr;

base_addr = fmc_bank0_base_address_get( );
```

### fmc\_bank1\_base\_address\_get

The description of fmc\_bank1\_base\_address\_get is shown as below:

**Table 3-481. Function fmc\_bank1\_base\_address\_get**

<b>Function name</b>	fmc_bank1_base_address_get
<b>Function prototype</b>	uint32_t fmc_bank1_base_address_get(void);
<b>Function descriptions</b>	get base address of bank1. It is only valid when DBS=1
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	base address of bank1

Example:

```
/* get base address of bank1 */
uint32_t base_addr;

base_addr = fmc_bank1_base_address_get( );
```

### fmc\_page\_size\_get

The description of fmc\_page\_size\_get is shown as below:

**Table 3-482. Function fmc\_page\_size\_get**

<b>Function name</b>	fmc_page_size_get
<b>Function prototype</b>	uint32_t fmc_page_size_get(void);

<b>Function descriptions</b>	get page size
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	page size

Example:

```
/* get page size */
uint32_t page_size;
page_size = fmc_page_size_get( );
```

### fmc\_debugger\_enable

The description of fmc\_debugger\_enable is shown as below:

**Table 3-483. Function fmc\_debugger\_enable**

<b>Function name</b>	fmc_debugger_enable
<b>Function prototype</b>	void fmc_debugger_enable(void);
<b>Function descriptions</b>	enable debugger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable debugger */
fmc_debugger_enable( );
```

### fmc\_debugger\_disable

The description of fmc\_debugger\_disable is shown as below:

**Table 3-484. Function fmc\_debugger\_disable**

<b>Function name</b>	fmc_debugger_disable
<b>Function prototype</b>	void fmc_debugger_disable(void);

<b>Function descriptions</b>	disable debugger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable debugger */
```

```
fmc_debugger_disable( );
```

### fmc\_slp\_pd\_mode\_enable

The description of fmc\_slp\_pd\_mode\_enable is shown as below:

**Table 3-485. Function fmc\_slp\_pd\_mode\_enable**

<b>Function name</b>	fmc_slp_pd_mode_enable
<b>Function prototype</b>	void fmc_slp_pd_mode_enable(void);
<b>Function descriptions</b>	flash enter power down mode when MCU enter sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* flash enter power down mode when MCU enter sleep mode */
```

```
fmc_slp_pd_mode_enable( );
```

### fmc\_slp\_pd\_mode\_disable

The description of fmc\_slp\_pd\_mode\_disable is shown as below:

**Table 3-486. Function fmc\_slp\_pd\_mode\_disable**

<b>Function name</b>	fmc_slp_pd_mode_disable
<b>Function prototype</b>	void fmc_slp_pd_mode_disable (void);
<b>Function descriptions</b>	flash exit power down mode when MCU enter sleep mode



<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* flash enter idle mode when MCU enter sleep mode */
```

```
fmc_slp_pd_mode_disable( );
```

### fmc\_scr\_area\_enable

The description of fmc\_scr\_area\_enable is shown as below:

**Table 3-487. Function fmc\_scr\_area\_enable**

<b>Function name</b>	fmc_scr_area_enable
<b>Function prototype</b>	void fmc_scr_area_enable(uint32_t scr_area);
<b>Function descriptions</b>	enable secure area protection
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>scr_area</b>	secure user area
SCR_AREA0	the secure user area of bank0
SCR_AREA1	the secure user area of bank1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
fmc_unlock();
```

```
/* enable secure area protection */
```

```
fmc_scr_area_enable(SCR_AREA0);
```

### ob\_unlock

The description of ob\_unlock is shown as below:

**Table 3-488. Function ob\_unlock**

<b>Function name</b>	ob_unlock
----------------------	-----------

<b>Function prototype</b>	void ob_unlock(void);
<b>Function descriptions</b>	unlock the option bytes operation
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
fmc_unlock();

/* unlock the option bytes operation */

ob_unlock();
```

### ob\_lock

The description of ob\_lock is shown as below:

**Table 3-489. Function ob\_lock**

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

Example:

```
fmc_unlock();

/*lock the option bytes operation */

ob_lock();
```

### ob\_reload

The description of ob\_reload is shown as below:

Table 3-490. Function ob\_reload

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

Example:

```
fmc_unlock();

ob_unlock();

/* reload the option bytes operation */

ob_reload( );
```

### ob\_user\_write

The description of ob\_user\_write is shown as below:

Table 3-491. Function ob\_user\_write

<b>Function name</b>	ob_user_write
<b>Function prototype</b>	fmc_state_enum ob_user_write(uint32_t ob_user, uint32_t ob_user_mask);
<b>Function descriptions</b>	program the option bytes USER
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	fmc_ready_wait
<b>Input parameter{in}</b>	
<b>ob_user</b>	user option bytes. When write one paramter, the corresponding mask should be set.
<i>OB_NRST_PIN_INPUT_MODE</i>	a low level on the NRST pin can reset system, internal reset can not drive NRST pin
<i>OB_NRST_PIN_NORMAL_GPIO</i>	NRST pin function as normal GPIO
<i>OB_NRST_PIN_INPUT_OUTPUT_MODE</i>	NRST pin configure as input/output mode
<i>OB_NBOOT0_VALUE_0</i>	option byte BOOT0 is value 1
<i>OB_NBOOT0_VALUE_1</i>	option byte BOOT0 is value 0

OB_NSWBOOT0_FRO M_OB_BOOT0	BOOT0 taken from the option bit NBOOT0
OB_NSWBOOT0_FRO M_PIN	BOOT0 taken from PB8/BOOT0 pin
OB_TCM_SRAM_ERAS E_ENABLE	TCM SRAM erased if a system reset occurs
OB_TCM_SRAM_ERAS E_DISABLE	TCM SRAM is not erased if a system reset occurs
OB_SRAM_ECC_ENA BLE	SRAM and TCM SRAM ECC enable
OB_SRAM_ECC_DISA BLE	SRAM and TCM SRAM ECC disable
OB_NBOOT1_VALUE_ 0	option byte BOOT1 is value 1
OB_NBOOT1_VALUE_ 1	option byte BOOT1 is value 0
OB_SINGLE_BANK_M ODE	Single-bank
OB_DUAL_BANK_MO DE	Double-bank
OB_BB_DISABLE	boot from bank0
OB_BB_ENABLE	boot from bank1 or bank0 if bank1 is void
OB_STDBY_FWDGT_ SUSPEND	free watchdog is suspended in standby mode
OB_STDBY_FWDGT_ RUN	free watchdog is running in standby mode
OB_DPSLP_FWDGT_ SUSPEND	free watchdog is suspended in deep-sleep mode
OB_DPSLP_FWDGT_ RUN	free watchdog is running in deep-sleep mode
OB_FWDGT_HW	hardware free watchdog
OB_FWDGT_SW	software free watchdog
OB_STDBY_RST	generate a reset instead of entering standby mode
OB_STDBY_NRST	no reset when entering deepsleep mode
OB_DEEPSLEEP_RST	generate a reset instead of entering deepsleep mode
OB_DEEPSLEEP_NRS T	no reset when entering deepsleep mode
OB_BOR_TH_VALUE0	BOR threshold value 0
OB_BOR_TH_VALUE1	BOR threshold value 1
OB_BOR_TH_VALUE2	BOR threshold value 2
OB_BOR_TH_VALUE3	BOR threshold value 3
<b>Input parameter{in}</b>	
ob_user_mask	user bits mask

<i>FMC_OBCTL_NRST_</i> <i>MDSEL</i>	reset pin mode bit
<i>FMC_OBCTL_NBOOT</i> <i>0</i>	NBOOT0 option bit
<i>FMC_OBCTL_NSWBT</i> <i>0</i>	software BOOT0 bit
<i>FMC_OBCTL_TCMSR</i> <i>AM_ERS</i>	TCM SRAM erase while system reset bit
<i>FMC_OBCTL_SRAM_</i> <i>ECCEN</i>	SRAM and TCM SRAM ECC disable bit
<i>FMC_OBCTL_NBOOT</i> <i>1</i>	NBOOT1 option bit
<i>FMC_OBCTL_DBS</i>	bank mode bit
<i>FMC_OBCTL_BB</i>	dual-bank boot bit
<i>FMC_OBCTL_FWDGS</i> <i>PD_STDBY</i>	FWDGT suspend option in standby mode configuration bit
<i>FMC_OBCTL_FWDGS</i> <i>PD_DPSLP</i>	FWDGT suspend option in deepsleep mode configuration bit
<i>FMC_OBCTL_NFWDG</i> <i>_HW</i>	free watchdog configuration bit
<i>FMC_OBCTL_NRST_S</i> <i>TDBY</i>	option byte standby reset value bit
<i>FMC_OBCTL_NRST_D</i> <i>PSLP</i>	option byte deepsleep reset value bit
<i>FMC_OBCTL_BOR_TH</i>	BOR threshold status bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* modify option byte */

fmc_state = ob_user_write (OB_BB_ENABLE, FMC_OBCTL_BB);
```

### ob\_security\_protection\_config

The description of ob\_security\_protection\_config is shown as below:

Table 3-492. Function ob\_security\_protection\_config

Function name	ob_security_protection_config
Function prototype	fmc_state_enum ob_security_protection_config(uint8_t ob_spc);
Function descriptions	configure security protection
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
ob_spc	specify security protection
FMC_NSPC	no security protection
FMC_LSPC	low security protection
FMC_HSPC	high security protection
Output parameter{out}	
-	-
Return value	
fmc_state_enum	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* disable security protection */

fmc_state = ob_security_protection_config(FMC_NSPC);
```

## ob\_dcrp\_config

The description of ob\_dcrp\_config is shown as below:

Table 3-493. Function ob\_dcrp\_config

Function name	ob_dcrp_config
Function prototype	fmc_state_enum ob_dcrp_config(uint32_t dcrp_area, uint32_t dcrp_eren, uint32_t dcrp_start_addr, uint32_t dcrp_end_addr);
Function descriptions	configure the option byte DCRP area
Precondition	ob_unlock
The called functions	fmc_ready_wait
Input parameter{in}	
dcrp_area	DCRP area
DCRP_AREA0	the DCRP area of bank0 when DBS=1 or the first area of whole memory when DBS=0
DCRP_AREA1	the DCRP area of bank1 when DBS=1 or the second area of whole memory when DBS=0
Input parameter{in}	
dcrp_eren	DCRP area erase enable bit

<i>OB_DCRP_AREA_ERASE_DISABLE</i>	DCRP area is not erased when low security protection to no security protection
<i>OB_DCRP_AREA_ERASE_ENABLE</i>	DCRP area is erased when low security protection to no security protection
<b>Input parameter{in}</b>	
<b>dcrp_start</b>	first doubleword of DCRP area address when DBS=1 or first 2 x doubleword of DCRP area when DBS=0
<b>Input parameter{in}</b>	
<b>dcrp_end</b>	last doubleword of DCRP area address when DBS=1 or last 2 x doubleword of DCRP area when DBS=0
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* configure DCRP area */
```

```
fmc_state = fmc_state_enum ob_dcrp_config(DCRP_AREA0, OB_DCRP_AREA_ERASE_ENABLE, 0x08002000, 0x08002400)
```

### ob\_write\_protection\_config

The description of ob\_write\_protection\_config is shown as below:

**Table 3-494. Function ob\_write\_protection\_config**

<b>Function name</b>	ob_write_protection_config
<b>Function prototype</b>	fmc_state_enum ob_write_protection_config(uint32_t wp_area, uint32_t wp_start, uint32_t wp_end);
<b>Function descriptions</b>	configure the option byte write protection area
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	fmc_ready_wait
<b>Input parameter{in}</b>	
<b>wp_area</b>	write protection area
<i>BK0WP_AREA0</i>	the first area of bank0 when DBS=1 or the first area of whole memory when DBS=0
<i>BK0WP_AREA1</i>	the second area of bank0 when DBS=1 or the second area of whole memory when DBS=0
<i>BK1WP_AREA0</i>	the first area of bank1 when DBS=1 or the third area of whole memory when DBS=0

<i>BK1WP_AREA1</i>	the second area of bank1 when DBS=1 or the fourth area of whole memory when DBS=0
<b>Input parameter{in}</b>	
<b>wp_start</b>	first page of write protection area
<b>Input parameter{in}</b>	
<b>wp_end</b>	last page of write protection area
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* configure write protection area */
```

```
fmc_state = fmc_state_enum ob_write_protection_config(BK0WP_AREA0, 0x08, 0x09);
```

### ob\_scr\_area\_config

The description of ob\_scr\_area\_config is shown as below:

**Table 3-495. Function ob\_scr\_area\_config**

<b>Function name</b>	ob_scr_area_config
<b>Function prototype</b>	fmc_state_enum ob_scr_area_config(uint32_t scr_area, uint32_t secure_size);
<b>Function descriptions</b>	configure the option byte secure area
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	fmc_ready_wait
<b>Input parameter{in}</b>	
<b>scr_area</b>	secure area
<i>SCR_AREA0</i>	the secure area of bank0
<i>SCR_AREA1</i>	the secure area of bank1. It is useful only in dual bank mode.
<b>Input parameter{in}</b>	
<b>secure_size</b>	size of secure area in page unit
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();
```



```
ob_unlock();

/* configure the option byte secure area */

fmc_state_enum fmc_state = ob_scr_area_config (SCR_AREA0, 0x02);
```

### ob\_boot\_lock\_config

The description of ob\_boot\_lock\_config is shown as below:

**Table 3-496. Function ob\_boot\_lock\_config**

<b>Function name</b>	ob_boot_lock_config
<b>Function prototype</b>	fmc_state_enum ob_boot_lock_config(uint32_t boot_config);
<b>Function descriptions</b>	配置boot锁定
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	fmc_ready_wait
<b>Input parameter{in}</b>	
<b>boot_config</b>	boot configuration
OB_BOOT_LOCK_FR OM_MAIN_FLASH	boot from main flash
OB_BOOT_UNLOCK	unlock boot
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();

ob_unlock();

/* unlock boot */

fmc_state_enum fmc_state = ob_boot_lock_config(OB_BOOT_UNLOCK);
```

### ob\_bank\_memory\_swap\_config

The description of ob\_bank\_memory\_swap\_config is shown as below:

**Table 3-497. Function ob\_bank\_memory\_swap\_config**

<b>Function name</b>	ob_bank_memory_swap_config
<b>Function prototype</b>	fmc_state_enum ob_bank_memory_swap_config(uint32_t swp_config);
<b>Function descriptions</b>	configure FMC memory mapping swap bit
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	fmc_ready_wait
<b>Input parameter{in}</b>	
<b>swp_config</b>	bank memory mapping swap configuration

<code>OB_BANK_MAPPING_SWAP</code>	swap bank memory mapping
<code>OB_BANK_MAPPING_NOT_SWAP</code>	do not swap bank memory mapping
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<code>fmc_state_enum</code>	FMC status, refers to <a href="#">Table 3-463. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

/* swap bank memory mapping */

fmc_state = ob_bank_memory_swap_config (OB_BANK_MAPPING_SWAP);
```

### ob\_user\_get

The description of ob\_user\_get is shown as below:

**Table 3-498. Function ob\_user\_get**

<b>Function name</b>	ob_user_get
<b>Function prototype</b>	uint32_t ob_user_get(void);
<b>Function descriptions</b>	get the value of option bytes USER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<code>uint32_t</code>	the option bytes USER value

Example:

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

uint32_t user_value;

user_value = ob_user_get();
```

### ob\_security\_protection\_level\_get

The description of ob\_security\_protection\_level\_get is shown as below:

**Table 3-499. Function ob\_security\_protection\_level\_get**

<b>Function name</b>	ob_security_protection_level_get
<b>Function prototype</b>	uint8_t ob_security_protection_level_get(void);

<b>Function descriptions</b>	get the value of FMC option bytes security protection level in FMC_OBCTL register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint8_t</b>	protection level
<i>FMC_NSPC</i>	no protection
<i>FMC_LSPC</i>	protection level low
<i>FMC_HSPC</i>	protection level high

Example:

```
/* get the value of FMC option bytes security protection level */
```

```
uint8_t user = ob_security_protection_level_get();
```

### ob\_dcrp\_area\_get

The description of ob\_dcrp\_area\_get is shown as below:

**Table 3-500. Function ob\_dcrp\_area\_get**

<b>Function name</b>	ob_dcrp_area_get
<b>Function prototype</b>	uint32_t ob_dcrp_area_get(uint32_t dcrp_area, uint32_t *dcrp_erase_option, uint32_t *dcrp_area_start_addr, uint32_t *dcrp_area_end_addr);
<b>Function descriptions</b>	get configuration of DCRP area
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dcrp_area</b>	DCRP area
<i>DCRP_AREA0</i>	the DCRP area of bank0 when DBS=1 or the first area of whole memory when DBS=0
<i>DCRP_AREA1</i>	the DCRP area of bank1 when DBS=1 or the second area of whole memory when DBS=0
<b>Output parameter{out}</b>	
<b>dcrp_erase_option</b>	erase option of DCRP area
<b>Output parameter{out}</b>	
<b>dcrp_start_addr</b>	start address of DCRP area
<b>Output parameter{out}</b>	
<b>dcrp_end_addr</b>	end address of DCRP area
<b>Return value</b>	

<b>uint32_t</b>	state of address
<i>INVLD_AREA_ADDRE</i> SS	the area address is invalid
<i>VLD_AREA_ADDRESS</i>	the area address is valid

Example:

```
/* get configuration of DCRP area */
```

```
uint32_t start, end, erase ,flag;
```

```
flag = ob_dcrp_area_get(DCRP_AREA0, &erase, &start, &end)
```

### ob\_write\_protection\_get

The description of ob\_write\_protection\_get is shown as below:

**Table 3-501. Function ob\_write\_protection\_get**

<b>Function name</b>	ob_write_protection_get
<b>Function prototype</b>	uint32_t ob_write_protection_get(uint32_t wp_area, uint32_t *wp_area_start_addr, uint32_t *wp_area_end_addr);
<b>Function descriptions</b>	get address of write protection area
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wp_area</b>	write protection area
<i>BK0WP_AREA0</i>	the first area of bank0 when DBS=1 or the first area of whole memory when DBS=0
<i>BK0WP_AREA1</i>	the second area of bank0 when DBS=1 or the second area of whole memory when DBS=0
<i>BK1WP_AREA0</i>	the first area of bank1 when DBS=1 or the third area of whole memory when DBS=0
<i>BK1WP_AREA1</i>	the second area of bank1 when DBS=1 or the fourth area of whole memory when DBS=0
<b>Output parameter{out}</b>	
<b>wp_area_start_addr</b>	start address of write protection area
<b>Output parameter{out}</b>	
<b>wp_area_end_addr</b>	end address of write protection area
<b>Return value</b>	
<b>uint32_t</b>	state of address
<i>INVLD_AREA_ADDRE</i> SS	the area address is invalid
<i>VLD_AREA_ADDRESS</i>	the area address is valid

Example:

```
/* get configuration of WP area */
```

```
uint32_t start, end, flag;
```

```
flag = ob_write_protection_get(BK0WP_AREA0, &start, &end);
```

### ob\_scr\_area\_size\_get

The description of ob\_scr\_area\_size\_get is shown as below:

**Table 3-502. Function ob\_scr\_area\_size\_get**

<b>Function name</b>	ob_scr_area_size_get
<b>Function prototype</b>	uint32_t ob_scr_area_size_get(uint32_t scr_area, uint32_t *scr_area_byte_cnt);
<b>Function descriptions</b>	get size of secure area
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>scr_area</b>	secure area
SCR_AREA0	the secure area of bank0
SCR_AREA1	the secure area of bank1
<b>Output parameter{out}</b>	
<b>secure_size</b>	secure area size in byte unit
<b>Return value</b>	
<b>uint32_t</b>	state of size
INVLD_AREA_ADDRES	the area size is invalid
VLD_AREA_ADDRESS	the area size is valid

Example:

```
/* get size of secure area */
```

```
uint32_t size, flag;
```

```
flag = ob_scr_area_size_get(SCR_AREA0, &size);
```

### ob\_boot\_config\_get

The description of ob\_boot\_config\_get is shown as below:

**Table 3-503. Function ob\_boot\_config\_get**

<b>Function name</b>	ob_boot_config_get
<b>Function prototype</b>	uint32_t ob_boot_config_get(void);
<b>Function descriptions</b>	get boot configuration
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-

Output parameter{out}	
-	-
Return value	
uint32_t	boot configuration

Example:

```
/* get boot value */
uint32_t boot_value;

boot_value = ob_boot_config_get();
```

### fmc\_flag\_get

The description of fmc\_flag\_get is shown as below:

**Table 3-504. Function fmc\_flag\_get**

Function name	fmc_flag_get
Function prototype	FlagStatus fmc_flag_get(uint32_t flag);
Function descriptions	get FMC flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	FMC flag
FMC_FLAG_BUSY	FMC operation is in progress
FMC_FLAG_OBERR	option byte read error
FMC_FLAG_WPERR	erase/program protection error
FMC_FLAG_PGSERR	program sequence error
FMC_FLAG_PGMERR	program size error
FMC_FLAG_PGAERR	program alignment error
FMC_FLAG_RPERR	read protection error
FMC_FLAG_PGERR	program error
FMC_FLAG_OPRERR	operation error
FMC_FLAG_ENDF	FMC end of operation flag
Output parameter{out}	
-	-
Return value	
FlagStatus	status of flag
RESET	reset
SET	set

Example:

```
/* get FMC flag status */

FlagStatus flag = fmc_flag_get(FMC_FLAG_ENDF);
```

## fmc\_flag\_clear

The description of fmc\_flag\_clear is shown as below:

**Table 3-505. Function fmc\_flag\_clear**

<b>Function name</b>	fmc_flag_clear
<b>Function prototype</b>	void fmc_flag_clear(uint32_t flag);
<b>Function descriptions</b>	clear the FMC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	FMC flag
<i>FMC_FLAG_OBERR</i>	option byte read error
<i>FMC_FLAG_WPERR</i>	erase/program protection error
<i>FMC_FLAG_PGSERR</i>	program sequence error
<i>FMC_FLAG_PGMERR</i>	program size error
<i>FMC_FLAG_PGAERR</i>	program alignment error
<i>FMC_FLAG_RPERR</i>	read protection error
<i>FMC_FLAG_PGERR</i>	program error
<i>FMC_FLAG_OPRERR</i>	operation error
<i>FMC_FLAG_ENDF</i>	FMC end of operation flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear FMC ENDF flag */
```

```
fmc_flag_clear(FMC_FLAG_ENDF);
```

## fmc\_interrupt\_enable

The description of fmc\_interrupt\_enable is shown as below:

**Table 3-506. Function fmc\_interrupt\_enable**

<b>Function name</b>	fmc_interrupt_enable
<b>Function prototype</b>	void fmc_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable FMC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	the FMC interrupt
<i>FMC_INT_END</i>	FMC end of operation interrupt
<i>FMC_INT_OPRERR</i>	FMC error interrupt

<i>FMC_INT_RPERR</i>	read protection error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FMC end interrupt */
fmc_interrupt_enable(FMC_INT_END);
```

### fmc\_interrupt\_disable

The description of fmc\_interrupt\_disable is shown as below:

**Table 3-507. Function fmc\_interrupt\_disable**

<b>Function name</b>	fmc_interrupt_disable
<b>Function prototype</b>	void fmc_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable FMC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	the FMC interrupt
<i>FMC_INT_END</i>	FMC end of operation interrupt
<i>FMC_INT_OPRERR</i>	FMC error interrupt
<i>FMC_INT_RPERR</i>	read protection error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FMC end interrupt */
fmc_interrupt_disable(FMC_INT_END);
```

### fmc\_ecc\_flag\_get

The description of fmc\_ecc\_flag\_get is shown as below:

**Table 3-508. Function fmc\_ecc\_flag\_get**

<b>Function name</b>	fmc_ecc_flag_get
<b>Function prototype</b>	FlagStatus fmc_ecc_flag_get(uint32_t flag);
<b>Function descriptions</b>	get ECC flag status
<b>Precondition</b>	-



<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	ECC flag
<i>FMC_FLAG_ECCDET0</i>	two-bit error detected flag(in 64bit LSB when DBS is 0)
<i>FMC_FLAG_ECCCOR0</i>	one-bit error detected and corrected flag(in 64bit LSB when DBS is 0)
<i>FMC_FLAG_ECCDET1</i>	two-bit error detected flag in 64bit MSB
<i>FMC_FLAG_ECCCOR1</i>	one-bit error detected and corrected flag in 64bit MSB
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	status of flag
<i>RESET</i>	reset
<i>SET</i>	set

Example:

```
/* get ECCDET0 flag status */
```

```
FlagStatus stat = fmc_ecc_flag_get(FMC_FLAG_ECCDET0);
```

### fmc\_ecc\_flag\_clear

The description of fmc\_ecc\_flag\_clear is shown as below:

**Table 3-509. Function fmc\_ecc\_flag\_clear**

<b>Function name</b>	fmc_ecc_flag_clear
<b>Function prototype</b>	void fmc_ecc_flag_clear (uint32_t flag);
<b>Function descriptions</b>	clear the ECC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	ECC flag
<i>FMC_FLAG_ECCDET0</i>	two-bit error detected flag(in 64bit LSB when DBS is 0)
<i>FMC_FLAG_ECCCOR0</i>	one-bit error detected and corrected flag(in 64bit LSB when DBS is 0)
<i>FMC_FLAG_ECCDET1</i>	two-bit error detected flag in 64bit MSB
<i>FMC_FLAG_ECCCOR1</i>	one-bit error detected and corrected flag in 64bit MSB
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear ECCDET0 flag status */
```

```
fmc_ecc_flag_clear(FMC_FLAG_ECCDET0);
```

### fmc\_ecccor\_interrupt\_enable

The description of fmc\_ecccor\_interrupt\_enable is shown as below:

**Table 3-510. Function fmc\_ecccor\_interrupt\_enable**

<b>Function name</b>	fmc_ecccor_interrupt_enable
<b>Function prototype</b>	void fmc_ecccor_interrupt_enable(void);
<b>Function descriptions</b>	enable ECCCOR interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ECCCOR interrupt */
```

```
fmc_ecccor_interrupt_enable();
```

### fmc\_ecccor\_interrupt\_disable

The description of fmc\_ecccor\_interrupt\_disable is shown as below:

**Table 3-511. Function fmc\_ecccor\_interrupt\_disable**

<b>Function name</b>	fmc_ecccor_interrupt_disable
<b>Function prototype</b>	void fmc_ecccor_interrupt_disable (void);
<b>Function descriptions</b>	disable ECCCOR interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ECCCOR interrupt */
```

```
fmc_eccor_interrupt_disable();
```

### fmc\_eccor\_interrupt\_flag\_get

The description of fmc\_eccor\_interrupt\_flag\_get is shown as below:

**Table 3-512. Function fmc\_eccor\_interrupt\_flag\_get**

<b>Function name</b>	fmc_eccor_interrupt_flag_get
<b>Function prototype</b>	FlagStatus fmc_eccor_interrupt_flag_get (uint32_t flag);
<b>Function descriptions</b>	get ECCOR interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	ECCOR interrupt flag
<i>FMC_INT_FLAG_ECCCOR0</i>	one-bit error detected and corrected flag
<i>FMC_INT_FLAG_ECCCOR1</i>	one-bit error detected and corrected flag in 64bit MSB
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	status of flag
<i>RESET</i>	reset
<i>SET</i>	set

Example:

```
/* get ECCDET0 interrupt flag status */
```

```
FlagStatus stat = fmc_eccor_interrupt_flag_get(FMC_INT_FLAG_ECCCOR0);
```

### fmc\_eccor\_interrupt\_flag\_clear

The description of fmc\_eccor\_interrupt\_flag\_clear is shown as below:

**Table 3-513. Function fmc\_eccor\_interrupt\_flag\_clear**

<b>Function name</b>	fmc_eccor_interrupt_flag_clear
<b>Function prototype</b>	void fmc_eccor_interrupt_flag_clear(uint32_t flag);
<b>Function descriptions</b>	clear ECCOR interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	ECCOR interrupt flag
<i>FMC_INT_FLAG_ECCCOR0</i>	one-bit error detected and corrected flag
<i>FMC_INT_FLAG_ECCCOR1</i>	one-bit error detected and corrected flag in 64bit MSB

<i>COR1</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear ECCDET0 interrupt flag status */
```

```
fmc_ecccor_interrupt_flag_clear(FMC_INT_FLAG_ECCCOR0);
```

### fmc\_interrupt\_flag\_get

The description of fmc\_interrupt\_flag\_get is shown as below:

**Table 3-514. Function fmc\_interrupt\_flag\_get**

<b>Function name</b>	fmc_interrupt_flag_get
<b>Function prototype</b>	FlagStatus fmc_interrupt_flag_get(fmc_interrupt_flag_enum flag);
<b>Function descriptions</b>	get FMC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>interrupt</b>	FMC interrupt flag
<i>FMC_INT_END</i>	FMC end of operation interrupt flag
<i>FMC_INT_ERR</i>	FMC error interrupt flag
<i>FMC_INT_RPERR</i>	read protection error interrupt flag
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	status of flag
<i>RESET</i>	reset
<i>SET</i>	set

Example:

```
/* get FMC RPERR error flag */
```

```
FlagStatus flag = fmc_interrupt_flag_get(FMC_INT_FLAG_RPERR);
```

### fmc\_interrupt\_flag\_clear

The description of fmc\_interrupt\_flag\_clear is shown as below:

**Table 3-515. Function fmc\_interrupt\_flag\_clear**

<b>Function name</b>	fmc_interrupt_flag_clear
<b>Function prototype</b>	void fmc_interrupt_flag_clear(uint32_t flag);

<b>Function descriptions</b>	clear FMC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FMC interrupt flag
<i>FMC_INT_END</i>	FMC end of operation interrupt flag
<i>FMC_INT_ERR</i>	FMC error interrupt flag
<i>FMC_INT_RPERR</i>	read protection error interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear FMC RPERR error flag */
```

```
fmc_interrupt_flag_clear(FMC_INT_FLAG_RPERR);
```

### fmc\_pd\_mode\_enter

The description of fmc\_pd\_mode\_enter is shown as below:

**Table 3-516. Function fmc\_pd\_mode\_enter**

<b>Function name</b>	fmc_pd_mode_enter
<b>Function prototype</b>	__attribute__((section("RAMCODE"))) void fmc_pd_mode_enter(void);
<b>Function descriptions</b>	flash enter power down mode when MCU run mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* flash enter power down mode when MCU run mode */
```

```
fmc_pd_mode_enter();
```

### fmc\_pd\_mode\_exit

The description of fmc\_pd\_mode\_exit is shown as below:

Table 3-517. Function `fmc_pd_mode_exit`

Function name	<code>fmc_pd_mode_exit</code>
Function prototype	<code>__attribute__((section("RAMCODE"))) void fmc_pd_mode_exit(void);</code>
Function descriptions	flash exit power down mode when MCU run mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* flash exit power down mode when MCU run mode */
```

```
fmc_pd_mode_exit();
```

### `ob_bank_mode_config`

The description of `ob_bank_mode_config` is shown as below:

Table 3-518. Function `ob_bank_mode_config`

Function name	<code>ob_bank_mode_config</code>
Function prototype	<code>__attribute__((section("RAMCODE"))) ErrStatus ob_bank_mode_config(uint32_t bank_mode);</code>
Function descriptions	configure the option byte bank mode
Precondition	-
The called functions	-
Input parameter{in}	
<code>bank_mode</code>	bank mode
<code>OB_SINGLE_BANK_MODE</code>	single bank mode
<code>OB_DUAL_BANK_MODE</code>	dual bank mode
Output parameter{out}	
-	-
Return value	
<code>ErrStatus</code>	ERROR / SUCCESS

Example:

```
/* configure the option byte to single bank mode */
```

```
ErrStatus flag;
```

```
flag = ob_bank_mode_config(OB_SINGLE_BANK_MODE);
```

## 3.17. 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 [3.17.1](#). The FWDGT firmware functions are introduced in chapter [3.17.2](#).

### 3.17.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

**Table 3-519. FWDGT Registers**

Registers	Descriptions
FWDGT_CTL	control register
FWDGT_PSC	prescaler register
FWDGT_RLD	reload register
FWDGT_STAT	status register
FWDGT_WND	window register

### 3.17.2. Descriptions of Peripheral functions

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

**Table 3-520. 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 3-521. Function fwdgt\_write\_enable

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

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 3-522. Function fwdgt\_write\_disable

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

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 3-523. 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 3-524. 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 3-525. Function fwdgt\_reload\_value\_config

<b>Function name</b>	fwdgt_reload_value_config
<b>Function prototype</b>	ErrStatus fwdgt_reload_value_config(uint16_t reload_value);
<b>Function descriptions</b>	configure the FWDGT counter reload value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>reload_value</b>	reload_value, specify reload value(0x0000 - 0x0FFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR / SUCCESS

Example:

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

```
ErrStatus flag;
```

```
flag = fwdgt_reload_value_config(0xFFFF);
```

### fwdgt\_window\_value\_config

The description of fwdgt\_window\_value\_config is shown as below:

Table 3-526. Function fwdgt\_window\_value\_config

<b>Function name</b>	fwdgt_window_value_config
<b>Function prototype</b>	ErrStatus fwdgt_window_value_config(uint16_t window_value);
<b>Function descriptions</b>	configure the FWDGT counter window value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>window_value</b>	window_value, specify window value(0x0000 - 0x0FFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR / SUCCESS

Example:

```
/* set FWDGT window value to 0xFFFF */
```

```
ErrStatus flag;
```

```
flag = fwdgt_window_value_config(0xFFFF);
```

## fwdgt\_counter\_reload

The description of fwdgt\_counter\_reload is shown as below:

**Table 3-527. Function fwdgt\_counter\_reload**

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

Example:

```
/* reload FWDGT counter */
```

```
fwdgt_counter_reload();
```

## fwdgt\_config

The description of fwdgt\_config is shown as below:

**Table 3-528. Function fwdgt\_config**

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

Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* configure FWDGT counter clock: 32KHz(IRC32K) / 64 = 0.5 KHz */
```

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

### fwdgt\_flag\_get

The description of fwdgt\_flag\_get is shown as below:

**Table 3-529. 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
FWDGT_FLAG_PUD	a write operation to FWDGT_PSC register is on going
FWDGT_FLAG_RUD	a write operation to FWDGT_RLD register is on going
FWDGT_FLAG_WUD	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);
```

## 3.18. GPIO

GPIO is used to implement logic input/output functions for the devices. The GPIO registers are listed in chapter [3.18.1](#), the GPIO firmware functions are introduced in chapter [3.18.2](#).

### 3.18.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

**Table 3-530. GPIO Registers**

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

### 3.18.2. Descriptions of Peripheral functions

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

**Table 3-531. 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_filter_set	set GPIO input filter
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 3-532. Function gpio\_deinit

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

Example:

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

## gpio\_mode\_set

The description of gpio\_mode\_set is shown as below:

Table 3-533. Function gpio\_mode\_set

<b>Function name</b>	gpio_mode_set
<b>Function prototype</b>	void gpio_mode_set(uint32_t gpio_periph, uint32_t mode, uint32_t pull_up_down, uint32_t pin);
<b>Function descriptions</b>	set GPIO mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Input parameter{in}</b>	
<b>mode</b>	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
<b>Input parameter{in}</b>	
<b>pull_up_down</b>	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_PULLDO</i> <i>WN</i>	with pull-down resistor
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-534. Function gpio\_output\_options\_set**

<b>Function name</b>	gpio_output_options_set
<b>Function prototype</b>	void gpio_output_options_set(uint32_t gpio_periph, uint8_t otype, uint32_t speed, uint32_t pin);
<b>Function descriptions</b>	set GPIO output type and speed
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Input parameter{in}</b>	
<b>otype</b>	gpio pin output mode
<i>GPIO_OTYPE_PP</i>	push pull mode
<i>GPIO_OTYPE_OD</i>	open drain mode
<b>Input parameter{in}</b>	
<b>speed</b>	gpio pin output max speed
<i>GPIO_OSPEED_12MH</i> <i>Z</i>	output max speed 12MHz
<i>GPIO_OSPEED_60MH</i> <i>Z</i>	output max speed 60MHz
<i>GPIO_OSPEED_85MH</i> <i>Z</i>	output max speed 85MHz
<i>GPIO_OSPEED_100_2</i> <i>20MHZ</i>	output max speed 100/220MHz

Input parameter{in}	
<b>pin</b>	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_12MHZ,
GPIO_PIN_0);
```

### gpio\_bit\_set

The description of gpio\_bit\_set is shown as below:

**Table 3-535. Function gpio\_bit\_set**

<b>Function name</b>	gpio_bit_set
<b>Function prototype</b>	void gpio_bit_set(uint32_t gpio_periph,uint32_t pin);
<b>Function descriptions</b>	set GPIO pin bit
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
Input parameter{in}	
<b>pin</b>	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 3-536. Function gpio\_bit\_reset

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

Example:

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

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

## gpio\_bit\_write

The description of gpio\_bit\_write is shown as below:

Table 3-537. Function gpio\_bit\_write

<b>Function name</b>	gpio_bit_write
<b>Function prototype</b>	void gpio_bit_write(uint32_t gpio_periph,uint32_t pin,bit_status bit_value);
<b>Function descriptions</b>	write data to the specified GPIO pin
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Input parameter{in}</b>	
<b>bit_value</b>	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 3-538. 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,G)
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\_filter\_set

The description of gpio\_input\_filter\_set is shown as below:

**Table 3-539. Function gpio\_input\_filter\_set**

Function name	gpio_input_filter_set
Function prototype	void gpio_input_filter_set(uint32_t gpio_periph, uint8_t speriod, uint32_t iftype, uint32_t pin);
Function descriptions	set GPIO input filter
Precondition	-

The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G)
Input parameter{in}	
speriod	gpio pin input sample period
GPIO_ISPERIOD(x)	period (x = 0 ~ 255)
Input parameter{in}	
iftype	gpio pin input filtering type
GPIO_IFTYPE_SYNC	input filter type synchronization
GPIO_IFTYPE_3_SAMPLES	input filter type filter 3 samples
GPIO_IFTYPE_6_SAMPLES	input filter type filter 6 samples
GPIO_IFTYPE_ASYNC	input filter type asynchronous
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 GPIO input filter */
```

```
gpio_input_filter_set(GPIOA, GPIO_ISPERIOD(100), GPIO_IFTYPE_SYNC);
```

### gpio\_input\_bit\_get

The description of gpio\_input\_bit\_get is shown as below:

**Table 3-540. 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,G)
Input parameter{in}	
pin	GPIO pin

<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-541. Function gpio\_input\_port\_get**

<b>Function name</b>	gpio_input_port_get
<b>Function prototype</b>	uint16_t gpio_input_port_get(uint32_t gpio_periph);
<b>Function descriptions</b>	get GPIO port input status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	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 3-542. Function gpio\_output\_bit\_get**

<b>Function name</b>	gpio_output_bit_get
<b>Function prototype</b>	FlagStatus gpio_output_bit_get(uint32_t gpio_periph,uint32_t pin);
<b>Function descriptions</b>	get GPIO pin output status

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-543. Function gpio\_output\_port\_get**

<b>Function name</b>	gpio_output_port_get
<b>Function prototype</b>	uint16_t gpio_output_port_get(uint32_t gpio_periph);
<b>Function descriptions</b>	get GPIO port output status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>Uint16_t</b>	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 3-544. Function gpio\_af\_set**

<b>Function name</b>	gpio_af_set
<b>Function prototype</b>	void gpio_af_set(uint32_t gpio_periph, uint32_t alt_func_num, uint32_t pin);
<b>Function descriptions</b>	set GPIO alternate function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Input parameter{in}</b>	
<b>alt_func_num</b>	GPIO pin af function, please refer to specific device datasheet
<i>GPIO_AF_0</i>	<i>RTC, HPDF</i>
<i>GPIO_AF_1</i>	<i>TIMER0, TIMER1, LPTIMER, TIMER4, TIMER14, TIMER15, TIMER16</i>
<i>GPIO_AF_2</i>	<i>TIMER0, TIMER1, TIMER2, TIMER3, TIMER4, TIMER7, TIMER14, TIMER19, I2C2</i>
<i>GPIO_AF_3</i>	<i>CLA0, CLA1, CLA2, CLA3, I2C0, I2C3, TIMER7, TIMER14, TIMER19, HPDF, CMP2, HRTIMER</i>
<i>GPIO_AF_4</i>	<i>TIMER0, TIMER7, TIMER15, TIMER16, I2C0, I2C1, I2C2, I2C3, EXMC, HPDF</i>
<i>GPIO_AF_5</i>	<i>SPI0, SPI1, IFRP, TIMER7, HPDF, CLA3, UART3, UART4</i>
<i>GPIO_AF_6</i>	<i>UART3, SPI1, SPI2, TIMER0, TIMER4, TIMER7, TIMER19, HRTIMER, IFRP, HPDF</i>
<i>GPIO_AF_7</i>	<i>USART0, USART1, USART2, TIMER19, CMP4, CMP5, CMP6</i>
<i>GPIO_AF_8</i>	<i>CMP0, CMP1, CMP2, CMP3, CMP4, CMP5, CMP6, CMP7, UART3, UART4, I2C2, I2C3, HPDF</i>
<i>GPIO_AF_9</i>	<i>TIMER0, TIMER7, TIMER14, EXMC, CAN0, CAN1, HRTIMER, QSPI</i>
<i>GPIO_AF_10</i>	<i>TIMER1, TIMER2, TIMER3, TIMER7, TIMER16, QSPI, HPDF</i>
<i>GPIO_AF_11</i>	<i>EXMC, TIMER0, TIMER7, CAN2, LPTIMER</i>
<i>GPIO_AF_12</i>	<i>SHTIMER, SPI1, TIMER0, TRIGSEL, CLA2, EXMC</i>
<i>GPIO_AF_13</i>	<i>TRIGSEL, HRTIMER</i>
<i>GPIO_AF_14</i>	<i>TIMER1, TIMER14, UART3, UART4, TRIGSEL, EXMC, TLI</i>
<i>GPIO_AF_15</i>	<i>EVENTOUT</i>
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

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 3-545. Function gpio\_pin\_lock**

<b>Function name</b>	gpio_pin_lock
<b>Function prototype</b>	void gpio_pin_lock(uint32_t gpio_periph, uint32_t pin);
<b>Function descriptions</b>	lock GPIO pin bit
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* lock PA0 */
```

```
gpio_pin_lock(GPIOA, GPIO_PIN_0);
```

## gpio\_bit\_toggle

The description of gpio\_bit\_toggle is shown as below:

**Table 3-546. Function gpio\_bit\_toggle**

<b>Function name</b>	gpio_bit_toggle
<b>Function prototype</b>	void gpio_bit_toggle(uint32_t gpio_periph, uint32_t pin);
<b>Function descriptions</b>	toggle GPIO pin status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G)
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	GPIO_PIN_ALL
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### gpio\_port\_toggle

The description of gpio\_port\_toggle is shown as below:

**Table 3-547. Function gpio\_port\_toggle**

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

Example:

```
/* toggle GPIOA */
```

```
gpio_port_toggle(GPIOA);
```

## 3.19. HPDF

A high performance digital filter module (HPDF) for external sigma delta ( $\Sigma$ - $\Delta$ ) modulator is integrated in GD32G5x3. The HPDF registers are listed in chapter [3.19.1](#), the HPDF firmware functions are introduced in chapter [3.19.2](#).



### 3.19.1. Descriptions of Peripheral registers

HPDF registers are listed in the table shown as below:

**Table 3-548. HPDF Registers**

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

### 3.19.2. Descriptions of Peripheral functions

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

**Table 3-549. HPDF firmware function**

Function name	Function description
hpdf_deinit	reset HPDF
hpdf_channel_struct_para_init	initialize the parameters of HPDF channel struct with the default values
hpdf_filter_struct_para_init	initialize the parameters of HPDF filter struct with the default values
hpdf_rc_struct_para_init	initialize the parameters of regular conversion struct with the default values
hpdf_ic_struct_para_init	initialize the parameters of inserted conversion struct with the default values

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

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

Function name	Function description
hpdf_flag_clear	clear the HPDF flags
hpdf_interrupt_enable	enable HPDF interrupt
hpdf_interrupt_disable	disable HPDF interrupt
hpdf_interrupt_flag_get	get the HPDF interrupt flags
hpdf_interrupt_flag_clear	clear the HPDF interrupt flags

### Structure hpdf\_channel\_parameter\_struct

**Table 3-550. Structure hpdf\_channel\_parameter\_struct**

Member name	Function description
data_packing_mode	ata packing mode for HPDF_CHxPDI register
channel_muxlexer	channel multiplexer select input data source
channel_pin_select	channel inputs pins selection
ck_loss_detector	clock loss detector
malfunction_monitor	malfunction monitor
spi_ck_source	SPI clock source select
serial_interface	serial interface type
calibration_offset	24-bit calibration offset
right_bit_shift	data right bit-shift
tm_filter	threshold monitor Sinc filter order selection
tm_filter_oversample	threshold monitor filter oversampling rate
mm_break_signal	malfunction monitor break signal distribution
mm_counter_threshold	malfunction monitor counter threshold
plsk_value	the number of serial input samples that will be skipped

### Structure hpdf\_filter\_parameter\_struct

**Table 3-551. Structure hpdf\_filter\_parameter\_struct**

Member name	Function description
tm_fast_mode	threshold monitor fast mode
tm_channel	threshold monitor channel
tm_high_threshold	threshold monitor high threshold
tm_low_threshold	threshold monitor low threshold value
extreme_monitor_channel	extremes monitor channel
sinc_filter	sinc filter order
sinc_oversample	sinc filter oversampling rate
integrator_oversample	integrator oversampling rate
ht_break_signal	high threshold event break signal distribution
lt_break_signal	low threshold event break signal distribution

## Structure hpdf\_rc\_parameter\_struct

**Table 3-552. Structure hpdf\_rc\_parameter\_struct**

Member name	Function description
fast_mode	fast conversion mode enable for regular conversions
rsc_channel	regular conversion channel
rcdmaen	DMA channel enabled to read data for the regular conversion
rcsyn	regular conversion synchronously
continuous_mode	regular conversions continuous mode

## Structure hpdf\_ic\_parameter\_struct

**Table 3-553. Structure hpdf\_ic\_parameter\_struct**

Member name	Function description
trigger_edge	inserted conversions trigger edge
trigger_signal	inserted conversions trigger signal
icdmaen	DMA channel enabled to read data for the inserted channel group
scmod	scan conversion mode of inserted conversions
icsyn	inserted conversion synchronously
ic_channel_group	inserted channel group selection

## Enum hpdf\_channel\_enum

**Table 3-554. Enum hpdf\_channel\_enum**

enum name	enum description
CHANNEL0	HPDF channel0
CHANNEL1	HPDF channel1
CHANNEL2	HPDF channel2
CHANNEL3	HPDF channel3
CHANNEL4	HPDF channel4
CHANNEL5	HPDF channel5
CHANNEL6	HPDF channel6
CHANNEL7	HPDF channel7

## Enum hpdf\_filter\_enum

**Table 3-555. Enum hpdf\_filter\_enum**

enum name	enum description
FLT0	HPDF filter0
FLT1	HPDF filter1
FLT2	HPDF filter2
FLT3	HPDF filter3

## Enum hpdf\_flag\_enum

**Table 3-556. Enum hpdf\_flag\_enum**

enum name	enum description
HPDF_FLAG_FLTY _ICEF	inserted conversion end flag
HPDF_FLAG_FLTY _RCEF	regular conversion end flag
HPDF_FLAG_FLTY _ICDOF	inserted conversion overflow flag
HPDF_FLAG_FLTY _RCDOF	regular conversion overflow flag
HPDF_FLAG_FLTY _TMEOF	threshold monitor event occurred flag
HPDF_FLAG_FLTY _ICPF	inserted conversion in progress flag
HPDF_FLAG_FLTY _RCPF	regular conversion in progress flag
HPDF_FLAG_FLT0 _CKLF0	clock loss on channel 0 flag
HPDF_FLAG_FLT0 _CKLF1	clock loss on channel 1 flag
HPDF_FLAG_FLT0 _CKLF2	clock loss on channel 2 flag
HPDF_FLAG_FLT0 _CKLF3	clock loss on channel 3 flag
HPDF_FLAG_FLT0 _CKLF4	clock loss on channel 4 flag
HPDF_FLAG_FLT0 _CKLF5	clock loss on channel 5 flag
HPDF_FLAG_FLT0 _CKLF6	clock loss on channel 6 flag
HPDF_FLAG_FLT0 _CKLF7	clock loss on channel 7 flag
HPDF_FLAG_FLT0 _MMF0	malfunction event occurred on channel 0 flag
HPDF_FLAG_FLT0 _MMF1	malfunction event occurred on channel 1 flag
HPDF_FLAG_FLT0 _MMF2	malfunction event occurred on channel 2 flag
HPDF_FLAG_FLT0 _MMF3	malfunction event occurred on channel 3 flag
HPDF_FLAG_FLT0	malfunction event occurred on channel 4 flag

enum name	enum description
_MMF4	
HPDF_FLAG_FLT0 _MMF5	malfunction event occurred on channel 5 flag
HPDF_FLAG_FLT0 _MMF6	malfunction event occurred on channel 6 flag
HPDF_FLAG_FLT0 _MMF7	malfunction event occurred on channel 7 flag
HPDF_FLAG_FLTY _RCHPDT	regular channel pending data
HPDF_FLAG_FLTY _LTF0	threshold monitor low threshold on channel 0 flag
HPDF_FLAG_FLTY _LTF1	threshold monitor low threshold on channel 1 flag
HPDF_FLAG_FLTY _LTF2	threshold monitor low threshold on channel 2 flag
HPDF_FLAG_FLTY _LTF3	threshold monitor low threshold on channel 3 flag
HPDF_FLAG_FLTY _LTF4	threshold monitor low threshold on channel 4 flag
HPDF_FLAG_FLTY _LTF5	threshold monitor low threshold on channel 5 flag
HPDF_FLAG_FLTY _LTF6	threshold monitor low threshold on channel 6 flag
HPDF_FLAG_FLTY _LTF7	threshold monitor low threshold on channel 7 flag
HPDF_FLAG_FLTY _HTF0	threshold monitor high threshold on channel 0 flag
HPDF_FLAG_FLTY _HTF1	threshold monitor high threshold on channel 1 flag
HPDF_FLAG_FLTY _HTF2	threshold monitor high threshold on channel 2 flag
HPDF_FLAG_FLTY _HTF3	threshold monitor high threshold on channel 3 flag
HPDF_FLAG_FLTY _HTF4	threshold monitor high threshold on channel 4 flag
HPDF_FLAG_FLTY _HTF5	threshold monitor high threshold on channel 5 flag
HPDF_FLAG_FLTY _HTF6	threshold monitor high threshold on channel 6 flag
HPDF_FLAG_FLTY _HTF7	threshold monitor high threshold on channel 7 flag

## Enum hpdf\_interrput\_flag\_enum

**Table 3-557. Enum hpdf\_interrput\_flag\_enum**

enum name	enum description
HPDF_INT_FLAG_FLTY_ICEF	inserted conversion end interrupt flag
HPDF_INT_FLAG_FLTY_RCEF	regular conversion end interruptflag
HPDF_INT_FLAG_FLTY_ICDOF	inserted conversion overflow interrupt flag
HPDF_INT_FLAG_FLTY_RCDOF	regular conversion overflow interrupt flag
HPDF_INT_FLAG_FLTY_TMEOF	threshold monitor event occurred interrupt flag
HPDF_INT_FLAG_FLT0_CKLF0	clock loss on channel 0 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF1	clock loss on channel 1 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF2	clock loss on channel 2 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF3	clock loss on channel 3 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF4	clock loss on channel 4 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF5	clock loss on channel 5 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF6	clock loss on channel 6 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF7	clock loss on channel 7 interrupt flag
HPDF_INT_FLAG_FLT0_MMF0	malfunction monitor detection on channel 0 interrupt flag
HPDF_INT_FLAG_FLT0_MMF1	malfunction monitor detection on channel 1 interrupt flag
HPDF_INT_FLAG_FLT0_MMF2	malfunction monitor detection on channel 2 interrupt flag
HPDF_INT_FLAG_FLT0_MMF3	malfunction monitor detection on channel 3 interrupt flag
HPDF_INT_FLAG_FLT0_MMF4	malfunction monitor detection on channel 4 interrupt flag
HPDF_INT_FLAG_FLT0_MMF5	malfunction monitor detection on channel 5 interrupt flag
HPDF_INT_FLAG_FLT0_MMF6	malfunction monitor detection on channel 6 interrupt flag



enum name	enum description
FLT0_MMF6	
HPDF_INT_FLAG_ FLT0_MMF7	malfunction monitor detection on channel 7 interrupt flag

## Enum hpdf\_interrput\_enum

**Table 3-558. Enum hpdf\_interrput\_enum**

enum name	enum description
HPDF_INT_FLTY_I CEIE	inserted conversion end interrupt enable
HPDF_INT_FLTY_R CEIE	regular conversion end interrupt enable
HPDF_INT_FLTY_I CDOIE	inserted conversion data overflow interrupt enable
HPDF_INT_FLTY_R CDOIE	regular conversion data overflow interrupt enable
HPDF_INT_FLTY_T MIE	threshold monitor interrupt enable
HPDF_INT_FLT0_M MIE	malfunction monitor interrupt enable
HPDF_INT_FLT0_C KLIE	clock loss interrupt enable

## hpdf\_deinit

The description of hpdf\_deinit is shown as below:

**Table 3-559. Function hpdf\_deinit**

<b>Function name</b>	hpdf_deinit
<b>Function prototype</b>	void hpdf_deinit(void);
<b>Function descriptions</b>	reset HPDF
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset HPDF */
```

```
hpdf_deinit();
```

## hpdf\_channel\_struct\_para\_init

The description of hpdf\_channel\_struct\_para\_init is shown as below:

**Table 3-560. Function hpdf\_channel\_struct\_para\_init**

<b>Function name</b>	hpdf_channel_struct_para_init
<b>Function prototype</b>	void hpdf_channel_struct_para_init(hpdf_channel_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of HPDF channel struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF channel, refer to <a href="#">Table 3-550. Structure hpdf_channel_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of HPDF channel */
hpdf_channel_parameter_struct hpdf_channel_init_struct;
hpdf_channel_struct_para_init(&hpdf_channel_init_struct);
```

## hpdf\_filter\_struct\_para\_init

The description of hpdf\_filter\_struct\_para\_init is shown as below:

**Table 3-561. Function hpdf\_filter\_struct\_para\_init**

<b>Function name</b>	hpdf_filter_struct_para_init
<b>Function prototype</b>	void hpdf_filter_struct_para_init(hpdf_filter_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of HPDF filter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF filter, refer to <a href="#">Table 3-551. Structure hpdf_filter_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of HPDF filter */
```

```
hpdf_filter_parameter_struct hpdf_filter_init_struct;
```

```
hpdf_filter_struct_para_init(&hpdf_filter_init_struct);
```

### hpdf\_rc\_struct\_para\_init

The description of hpdf\_rc\_struct\_para\_init is shown as below:

**Table 3-562. Function hpdf\_rc\_struct\_para\_init**

<b>Function name</b>	hpdf_rc_struct_para_init
<b>Function prototype</b>	void hpdf_rc_struct_para_init(hpdf_rc_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of regular conversion struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF regular conversion, refer to <a href="#">Table 3-552. Structure hpdf_rc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of regular conversion */
```

```
hpdf_rc_parameter_struct hpdf_rc_init_struct;
```

```
hpdf_rc_struct_para_init(&hpdf_rc_init_struct);
```

### hpdf\_ic\_struct\_para\_init

The description of hpdf\_ic\_struct\_para\_init is shown as below:

**Table 3-563. Function hpdf\_ic\_struct\_para\_init**

<b>Function name</b>	hpdf_ic_struct_para_init
<b>Function prototype</b>	void hpdf_ic_struct_para_init(hpdf_ic_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of inserted conversion struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF inserted conversion, refer to <a href="#">Table 3-553. Structure hpdf_ic_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* initialize the parameters of inserted conversion */
```

```
hpdf_ic_parameter_struct hpdf_ic_init_struct;
```

```
hpdf_ic_struct_para_init(&hpdf_ic_init_struct);
```

## hpdf\_enable

The description of hpdf\_enable is shown as below:

**Table 3-564. Function hpdf\_enable**

<b>Function name</b>	hpdf_enable
<b>Function prototype</b>	void hpdf_enable(void);
<b>Function descriptions</b>	enable the HPDF module globally
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the HPDF module */
```

```
hpdf_enable();
```

## hpdf\_disable

The description of hpdf\_disable is shown as below:

**Table 3-565. Function hpdf\_disable**

<b>Function name</b>	hpdf_disable
<b>Function prototype</b>	void hpdf_disable(void);
<b>Function descriptions</b>	disable the HPDF module globally
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable the HPDF module */

hpdf_disable();
```

## hpdf\_channel\_init

The description of hpdf\_channel\_init is shown as below:

**Table 3-566. Function hpdf\_channel\_init**

<b>Function name</b>	hpdf_channel_init
<b>Function prototype</b>	void hpdf_channel_init(hpdf_channel_enum channelx, hpdf_channel_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the HPDF channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF channel, refer to <a href="#">Table 3-550. Structure hpdf_channel_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the HPDF channel0 */

hpdf_channel_parameter_struct hpdf_channel_init_struct;

/* initialize HPDF channel0 */

hpdf_channel_init_struct.data_packing_mode = DPM_STANDARD_MODE;
hpdf_channel_init_struct.malfunction_monitor = MM_ENABLE;
hpdf_channel_init_struct.spi_ck_source = EXTERNAL_CKIN;
hpdf_channel_init_struct.channel_muxlexer = SERIAL_INPUT;
hpdf_channel_init_struct.serial_interface = SPI_RISING_EDGE;
hpdf_channel_init_struct.calibration_offset = 0;
hpdf_channel_init_struct.right_bit_shift = 0;
```

```

hpdf_channel_init_struct.mm_counter_threshold = 110;

hpdf_channel_init_struct.plsk_value = 0;

hpdf_channel_init(CHANNEL0, &hpdf_channel_init_struct);

```

## hpdf\_filter\_init

The description of hpdf\_filter\_init is shown as below:

**Table 3-567. Function hpdf\_filter\_init**

<b>Function name</b>	hpdf_filter_init
<b>Function prototype</b>	void hpdf_filter_init(hpdf_filter_enum filtery, hpdf_filter_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the HPDF filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..7)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF filter, refer to <a href="#">Table 3-551. Structure hpdf_filter_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* initialize the HPDF filter0 */

hpdf_filter_parameter_struct hpdf_filter_init_struct;

hpdf_filter_init_struct.tm_fast_mode = TMFM_DISABLE;

hpdf_filter_init_struct.tm_channel = TMCHEN_CHANNEL0;

hpdf_filter_init_struct.tm_high_threshold = tm_high_val;

hpdf_filter_init_struct.tm_low_threshold = tm_low_val;

hpdf_filter_init_struct.extreme_monitor_channel = EM_CHANNEL0;

hpdf_filter_init_struct.sinc_filter = FLT_SINC3;

hpdf_filter_init_struct.sinc_oversample = FLT_OVER_SAMPLE_32;

hpdf_filter_init_struct.integrator_oversample = INTEGRATOR_BYPASS;

hpdf_filter_init(FLT0, &hpdf_filter_init_struct);

```

## hpdf\_rc\_init

The description of hpdf\_rc\_init is shown as below:

**Table 3-568. Function hpdf\_rc\_init**

<b>Function name</b>	hpdf_rc_init
<b>Function prototype</b>	void hpdf_rc_init(hpdf_filter_enum filtery, hpdf_rc_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the regular conversion
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF regular conversion, refer to <a href="#">Table 3-552. Structure hpdf_rc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize regular conversion of the HPDF filter0 */
hpdf_rc_parameter_struct hpdf_rc_init_struct;
hpdf_rc_init_struct.fast_mode = FAST_DISABLE;
hpdf_rc_init_struct.rcs_channel = RCS_CHANNEL0;
hpdf_rc_init_struct.rcdmaen = RCDMAEN_ENABLE;
hpdf_rc_init_struct.continuous_mode = RCCM_ENABLE;
hpdf_rc_init(FLT0, &hpdf_rc_init_struct);
```

## hpdf\_ic\_init

The description of hpdf\_ic\_init is shown as below:

**Table 3-569. Function hpdf\_ic\_init**

<b>Function name</b>	hpdf_ic_init
<b>Function prototype</b>	void hpdf_ic_init(hpdf_filter_enum filtery, hpdf_ic_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the inserted conversion
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
Input parameter{in}	
*init_struct	the initialized struct needed to initialize HPDF inserted conversion, refer to <a href="#">Table 3-553. Structure hpdf_ic_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize inserted conversion of the HPDF filter0 */

hpdf_ic_parameter_struct hpdf_ic_init_struct;

hpdf_ic_init_struct.ic_channel_group = IGCSEL_CHANNEL0;

hpdf_ic_init_struct.scmmod = SCMOD_ENABLE;

hpdf_ic_init_struct.icdmaen = ICDMAEN_ENABLE;

hpdf_ic_init_struct.icsyn = ICSYN_DISABLE ;

hpdf_ic_init(FLT0, &hpdf_ic_init_struct);

```

### hpdf\_clock\_output\_config

The description of hpdf\_clock\_output\_config is shown as below:

**Table 3-570. Function hpdf\_clock\_output\_config**

Function name	hpdf_clock_output_config
Function prototype	void hpdf_clock_output_config(uint32_t source, uint8_t divider, uint32_t mode);
Function descriptions	configure serial output clock
Precondition	disable HPDF
The called functions	-
Input parameter{in}	
source	the HPDF serial clock output source
SERIAL_SYSTEM_CLK	serial clock output source is from system clock
SERIAL_AUDIO_CLK	serial clock output source is from audio clock
Input parameter{in}	
divider	serial clock output divider(0-255)
Input parameter{in}	
mode	serial clock output duty mode
CKOUTDM_DISABLE	disable serial clock output duty mode



<i>CKOUTDM_ENABLE</i>	enable serial clock output duty mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure serial clock output */
```

```
hpdf_clock_output_config(SERIAL_SYSTEM_CLK, 175, CKOUTDM_ENABLE);
```

### hpdf\_clock\_output\_source\_config

The description of hpdf\_clock\_output\_source\_config is shown as below:

**Table 3-571. Function hpdf\_clock\_output\_source\_config**

<b>Function name</b>	hpdf_clock_output_source_config
<b>Function prototype</b>	void hpdf_clock_output_source_config(uint32_t source);
<b>Function descriptions</b>	configure serial clock output source
<b>Precondition</b>	disable HPDF
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>source</b>	the HPDF serial clock output source
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from system clock
<i>SERIAL_AUDIO_CLK</i>	serial clock output source is from audio clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure serial clock output source */
```

```
hpdf_clock_output_config(SERIAL_SYSTEM_CLK);
```

### hpdf\_clock\_output\_duty\_mode\_disable

The description of hpdf\_clock\_output\_duty\_mode\_disable is shown as below:

**Table 3-572. Function hpdf\_clock\_output\_duty\_mode\_disable**

<b>Function name</b>	hpdf_clock_output_duty_mode_disable
<b>Function prototype</b>	void hpdf_clock_output_duty_mode_disable(void);
<b>Function descriptions</b>	disable serial clock output duty mode
<b>Precondition</b>	-
<b>The called functions</b>	-

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

Example:

```
/* disable serial clock output duty mode*/
```

```
hpdf_clock_output_duty_mode_disable();
```

### hpdf\_clock\_output\_duty\_mode\_enable

The description of hpdf\_clock\_output\_duty\_mode\_enable is shown as below:

**Table 3-573. Function hpdf\_clock\_output\_duty\_mode\_enable**

Function name	hpdf_clock_output_duty_mode_enable
Function prototype	void hpdf_clock_output_duty_mode_enable(void);
Function descriptions	enable serial clock output duty mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable serial clock output duty mode*/
```

```
hpdf_clock_output_duty_mode_enable();
```

### hpdf\_clock\_output\_divider\_config

The description of hpdf\_clock\_output\_divider\_config is shown as below:

**Table 3-574. Function hpdf\_clock\_output\_divider\_config**

Function name	hpdf_clock_output_divider_config
Function prototype	void hpdf_clock_output_divider_config(uint8_t divider);
Function descriptions	configure serial clock output divider
Precondition	disable HPDF
The called functions	-
Input parameter{in}	

<b>divider</b>	serial clock output divider(0-255)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure serial clock output divider */
hpdf_clock_output_divider_config (255);
```

### hpdf\_channel\_enable

The description of hpdf\_channel\_enable is shown as below:

**Table 3-575. Function hpdf\_channel\_enable**

<b>Function name</b>	hpdf_channel_enable
<b>Function prototype</b>	void hpdf_channel_enable(hpdf_channel_enum channelx);
<b>Function descriptions</b>	enable channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable channel0 */
hpdf_channel_enable(CHANNEL0);
```

### hpdf\_channel\_disable

The description of hpdf\_channel\_disable is shown as below:

**Table 3-576. Function hpdf\_channel\_disable**

<b>Function name</b>	hpdf_channel_disable
<b>Function prototype</b>	void hpdf_channel_disable(hpdf_channel_enum channelx);
<b>Function descriptions</b>	disable channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable channel0 */
```

```
hpdf_channel_disable(CHANNEL0);
```

### hpdf\_spi\_clock\_source\_config

The description of hpdf\_spi\_clock\_source\_config is shown as below:

**Table 3-577. Function hpdf\_spi\_clock\_source\_config**

<b>Function name</b>	hpdf_spi_clock_source_config
<b>Function prototype</b>	void hpdf_spi_clock_source_config(hpdf_channel_enum channelx, uint32_t clock_source);
<b>Function descriptions</b>	configure SPI clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>clock_source</b>	SPI clock source
<i>EXTERNAL_CKIN</i>	external input clock
<i>INTERNAL_CKOUT</i>	internal CKOUT clock
<i>HALF_CKOUT_FALLING_EDGE</i>	internal CKOUT clock, sampling point on each second CKOUT falling edge
<i>HALF_CKOUT_RISING_EDGE</i>	internal CKOUT clock, sampling point on each second CKOUT rising edge
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure external input clock as SPI clock source */
```

```
hpdf_spi_clock_source_config(CHANNEL0, EXTERNAL_CKIN);
```

## hpdf\_serial\_interface\_type\_config

The description of hpdf\_serial\_interface\_type\_config is shown as below:

**Table 3-578. Function hpdf\_serial\_interface\_type\_config**

<b>Function name</b>	hpdf_serial_interface_type_config
<b>Function prototype</b>	void hpdf_serial_interface_type_config(hpdf_channel_enum channelx, uint32_t type);
<b>Function descriptions</b>	configure serial interface type
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>type</b>	serial interface type
SPI_RISING_EDGE	SPI interface, sample data on rising edge
SPI_FALLING_EDGE	SPI interface, sample data on rising edge
MANCHESTER_CODE 0	Manchester coded input: rising edge = logic 0, falling edge = logic 1
MANCHESTER_CODE 1	Manchester coded input: rising edge = logic 1, falling edge = logic 0
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure external input clock as SPI clock source */
```

```
hpdf_spi_clock_source_config(CHANNEL0, EXTERNAL_CKIN);
```

## hpdf\_malfunction\_monitor\_disable

The description of hpdf\_malfunction\_monitor\_disable is shown as below:

**Table 3-579. Function hpdf\_malfunction\_monitor\_disable**

<b>Function name</b>	hpdf_malfunction_monitor_disable
<b>Function prototype</b>	void hpdf_malfunction_monitor_disable(hpdf_channel_enum channelx);
<b>Function descriptions</b>	disable malfunction monitor
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable malfunction monitor */
hpdf_malfunction_monitor_disable(CHANNEL0);
```

### hpdf\_malfunction\_monitor\_enable

The description of hpdf\_malfunction\_monitor\_enable is shown as below:

**Table 3-580. Function hpdf\_malfunction\_monitor\_enable**

Function name	hpdf_malfunction_monitor_enable
Function prototype	void hpdf_malfunction_monitor_enable(hpdf_channel_enum channelx);
Function descriptions	enable malfunction monitor
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable malfunction monitor */
hpdf_malfunction_monitor_enable(CHANNEL1);
```

### hpdf\_clock\_loss\_disable

The description of hpdf\_clock\_loss\_disable is shown as below:

**Table 3-581. Function hpdf\_clock\_loss\_disable**

Function name	hpdf_clock_loss_disable
Function prototype	void hpdf_clock_loss_disable(hpdf_channel_enum channelx);
Function descriptions	disable clock loss detector
Precondition	disable CHANNELx(x=0..7)
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module

<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable clock loss detector */
```

```
hpdf_clock_loss_disable(CHANNEL0);
```

### hpdf\_clock\_loss\_enable

The description of hpdf\_clock\_loss\_enable is shown as below:

**Table 3-582. Function hpdf\_clock\_loss\_enable**

<b>Function name</b>	hpdf_clock_loss_enable
<b>Function prototype</b>	void hpdf_clock_loss_enable(hpdf_channel_enum channelx);
<b>Function descriptions</b>	enable clock loss detector
<b>Precondition</b>	disable CHANNELx(x=0..7)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable clock loss detector */
```

```
hpdf_clock_loss_enable(CHANNEL0);
```

### hpdf\_channel\_pin\_redirection\_disable

The description of hpdf\_channel\_pin\_redirection\_disable is shown as below:

**Table 3-583. Function hpdf\_channel\_pin\_redirection\_disable**

<b>Function name</b>	hpdf_channel_pin_redirection_disable
<b>Function prototype</b>	void hpdf_channel_pin_redirection_disable(hpdf_channel_enum channelx);
<b>Function descriptions</b>	disable channel inputs pins redirection
<b>Precondition</b>	disable CHANNELx(x=0..7)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable channel0 inputs pins redirection */
```

```
hpdf_channel_pin_redirection_disable(CHANNEL0);
```

### hpdf\_channel\_pin\_redirection\_enable

The description of hpdf\_channel\_pin\_redirection\_enable is shown as below:

**Table 3-584. Function hpdf\_channel\_pin\_redirection\_enable**

<b>Function name</b>	hpdf_channel_pin_redirection_enable
<b>Function prototype</b>	void hpdf_channel_pin_redirection_enable(hpdf_channel_enum channelx);
<b>Function descriptions</b>	enable channel inputs pins redirection
<b>Precondition</b>	disable CHANNELx(x=0..7)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable channel0 inputs pins redirection */
```

```
hpdf_channel_pin_redirection_enable(CHANNEL0);
```

### hpdf\_channel\_multiplexer\_config

The description of hpdf\_channel\_multiplexer\_config is shown as below:

**Table 3-585. Function hpdf\_channel\_multiplexer\_config**

<b>Function name</b>	hpdf_channel_multiplexer_config
<b>Function prototype</b>	void hpdf_channel_multiplexer_config(hpdf_channel_enum channelx, uint32_t data_source);
<b>Function descriptions</b>	configure channel multiplexer select input data source
<b>Precondition</b>	disable CHANNELx(x=0..7)



<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>data_source</b>	input data source
<i>SERIAL_INPUT</i>	input data source is taken from serial inputs
<i>INTERNAL_INPUT</i>	input data source is taken from internal HPDF_CHxPDI register
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure channel multiplexer select input data source */
hpdf_channel_mux_config(CHANNEL0, SERIAL_INPUT);
```

### hpdf\_data\_pack\_mode\_config

The description of hpdf\_data\_pack\_mode\_config is shown as below:

**Table 3-586. Function hpdf\_data\_pack\_mode\_config**

<b>Function name</b>	hpdf_data_pack_mode_config
<b>Function prototype</b>	void hpdf_data_pack_mode_config(hpdf_channel_enum channelx, uint32_t mode);
<b>Function descriptions</b>	configure data packing mode
<b>Precondition</b>	disable CHANNELx(x=0..7)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>mode</b>	parallel data packing mode
<i>DPM_STANDARD_MODE</i>	standard mode
<i>DPM_INTERLEAVED_MODE</i>	interleaved mode
<i>DPM_DUAL_MODE</i>	dual mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data packing mode */
```

```
hpdf_data_pack_mode_config(CHANNEL0, DPM_STANDARD_MODE);
```

### hpdf\_data\_right\_bit\_shift\_config

The description of hpdf\_data\_right\_bit\_shift\_config is shown as below:

**Table 3-587. Function hpdf\_data\_right\_bit\_shift\_config**

<b>Function name</b>	hpdf_data_right_bit_shift_config
<b>Function prototype</b>	void hpdf_data_right_bit_shift_config(hpdf_channel_enum channelx, uint8_t right_shift);
<b>Function descriptions</b>	configure data right bit-shift
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>right_shift</b>	the number of bits that determine the right shift(0-31)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data right bit-shift */
```

```
hpdf_data_right_bit_shift_config(CHANNEL0, 5);
```

### hpdf\_calibration\_offset\_config

The description of hpdf\_calibration\_offset\_config is shown as below:

**Table 3-588. Function hpdf\_calibration\_offset\_config**

<b>Function name</b>	hpdf_calibration_offset_config
<b>Function prototype</b>	void hpdf_calibration_offset_config(hpdf_channel_enum channelx, int32_t offset);
<b>Function descriptions</b>	configure calibration offset
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	

<b>offset</b>	24-bit calibration offset, must be in (-8388608~8388607)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure calibration offset */
```

```
hpdf_calibration_offset_config (CHANNEL0, -255);
```

### hpdf\_malfunction\_break\_signal\_config

The description of hpdf\_malfunction\_break\_signal\_config is shown as below:

**Table 3-589. Function hpdf\_malfunction\_break\_signal\_config**

<b>Function name</b>	hpdf_malfunction_break_signal_config
<b>Function prototype</b>	void hpdf_malfunction_break_signal_config(hpdf_channel_enum channelx, uint32_t break_signal);
<b>Function descriptions</b>	configure malfunction monitor break signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>break_signal</b>	malfunction monitor break signal distribution
NO_MM_BREAK	break signal is not distributed to malfunction monitor on channel
MM_BREAK0	break signal 0 is distributed to malfunction monitor on channel
MM_BREAK1	break signal 1 is distributed to malfunction monitor on channel
MM_BREAK2	break signal 2 is distributed to malfunction monitor on channel
MM_BREAK3	break signal 3 is distributed to malfunction monitor on channel
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure break signal is distributed to malfunction monitor on channel0 */
```

```
hpdf_data_pack_mode_config(CHANNEL0, MM_BREAK0);
```

### hpdf\_malfunction\_counter\_config

The description of hpdf\_malfunction\_counter\_config is shown as below:

Table 3-590. Function `hpdf_malfunction_counter_config`

Function name	<code>hpdf_malfunction_counter_config</code>
Function prototype	<code>void hpdf_malfunction_counter_config(hpdf_channel_enum channelx, uint8_t threshold);</code>
Function descriptions	configure malfunction monitor counter threshold
Precondition	-
The called functions	-
Input parameter{in}	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum <code>hpdf_channel_enum</code></a>
Input parameter{in}	
<b>threshold</b>	malfunction monitor counter threshold(0-255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure malfunction monitor counter threshold */
hpdf_malfunction_counter_config(CHANNEL0, 255);
```

### `hpdf_write_parallel_data_standard_mode`

The description of `hpdf_write_parallel_data_standard_mode` is shown as below:

Table 3-591. Function `hpdf_write_parallel_data_standard_mode`

Function name	<code>hpdf_write_parallel_data_standard_mode</code>
Function prototype	<code>void hpdf_write_parallel_data_standard_mode(hpdf_channel_enum channelx, int16_t data);</code>
Function descriptions	write the parallel data on standard mode of data packing
Precondition	-
The called functions	-
Input parameter{in}	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum <code>hpdf_channel_enum</code></a>
Input parameter{in}	
<b>data</b>	the parallel data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write the parallel data on standard mode of data packing */
```

```
hpdf_write_parallel_data_standard_mode(CHANNEL0, 0xEFFF);
```

### hpdf\_write\_parallel\_data\_interleaved\_mode

The description of hpdf\_write\_parallel\_data\_interleaved\_mode is shown as below:

**Table 3-592. Function hpdf\_write\_parallel\_data\_interleaved\_mode**

<b>Function name</b>	hpdf_write_parallel_data_interleaved_mode
<b>Function prototype</b>	void hpdf_write_parallel_data_interleaved_mode(hpdf_channel_enum channelx, int32_t data);
<b>Function descriptions</b>	write the parallel data on interleaved mode of data packing
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>data</b>	the parallel data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write the parallel data on interleaved mode of data packing */
```

```
hpdf_write_parallel_data_interleaved_mode(CHANNEL0, 0xEFFFFFFF);
```

### hpdf\_write\_parallel\_data\_dual\_mode

The description of hpdf\_write\_parallel\_data\_dual\_mode is shown as below:

**Table 3-593. Function hpdf\_write\_parallel\_data\_dual\_mode**

<b>Function name</b>	hpdf_write_parallel_data_dual_mode
<b>Function prototype</b>	void hpdf_write_parallel_data_dual_mode(hpdf_channel_enum channelx, int32_t data);
<b>Function descriptions</b>	write the parallel data on dual mode of data packing
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	

<b>data</b>	the parallel data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write the parallel data on dual mode of data packing */
```

```
hpdf_write_parallel_data_dual_mode(CHANNEL0, 0xEFFFFFFF);
```

### hpdf\_pulse\_skip\_update

The description of hpdf\_pulse\_skip\_update is shown as below:

**Table 3-594. Function hpdf\_pulse\_skip\_update**

<b>Function name</b>	hpdf_pulse_skip_update
<b>Function prototype</b>	void hpdf_pulse_skip_update(hpdf_channel_enum channelx, uint8_t number);
<b>Function descriptions</b>	update the number of pulses to skip
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>number</b>	the number of serial input samples that will be skipped
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* update the number of pulses to skip */
```

```
pdf_pulse_skip_update(CHANNEL0, 63);
```

### hpdf\_pulse\_skip\_read

The description of hpdf\_pulse\_skip\_read is shown as below:

**Table 3-595. Function hpdf\_pulse\_skip\_read**

<b>Function name</b>	hpdf_pulse_skip_read
<b>Function prototype</b>	uint8_t hpdf_pulse_skip_read(hpdf_channel_enum channelx);
<b>Function descriptions</b>	read the number of pulses to skip

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint8_t</b>	the number of pulses to skip

Example:

```
/* read the number of pulses to skip */

uint8_t value;

value = hpdf_pulse_skip_read(CHANNEL0);
```

### hpdf\_filter\_enable

The description of hpdf\_filter\_enable is shown as below:

**Table 3-596. Function hpdf\_filter\_enable**

<b>Function name</b>	hpdf_filter_enable
<b>Function prototype</b>	void hpdf_filter_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable filter0 */

hpdf_filter_enable(FLT0);
```

### hpdf\_filter\_disable

The description of hpdf\_filter\_disable is shown as below:

**Table 3-597. Function hpdf\_filter\_disable**

<b>Function name</b>	hpdf_filter_disable
----------------------	---------------------

<b>Function prototype</b>	void hpdf_filter_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable filter0 */
```

```
hpdf_filter_disable(FLT0);
```

### hpdf\_filter\_config

The description of hpdf\_filter\_config is shown as below:

**Table 3-598. Function hpdf\_filter\_config**

<b>Function name</b>	hpdf_filter_config
<b>Function prototype</b>	void hpdf_filter_config(hpdf_filter_enum filtery, uint32_t order, uint16_t oversample);
<b>Function descriptions</b>	configure sinc filter order and oversample
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>order</b>	sinc filter order
<i>FLT_FASTSINC</i>	FastSinc filter type
<i>FLT_SINC1</i>	Sinc1 filter type
<i>FLT_SINC2</i>	Sinc2 filter type
<i>FLT_SINC3</i>	Sinc3 filter type
<i>FLT_SINC4</i>	Sinc4 filter type
<i>FLT_SINC5</i>	Sinc5 filter type
<b>Input parameter{in}</b>	
<b>oversample</b>	Sinc filter oversampling rate(1-1024)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	



-	-
---	---

Example:

```
/* configure sinc filter order and oversample */
```

```
hpdf_filter_config(FLT0, FLT_SINC2, 64);
```

### hpdf\_integrator\_oversample

The description of hpdf\_integrator\_oversample is shown as below:

**Table 3-599. Function hpdf\_integrator\_oversample**

<b>Function name</b>	hpdf_integrator_oversample
<b>Function prototype</b>	void hpdf_integrator_oversample(hpdf_filter_enum filtery, uint8_t oversample);
<b>Function descriptions</b>	configure integrator oversampling rate
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>oversample</b>	integrator oversampling rate(1-256)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure integrator oversampling rate */
```

```
hpdf_integrator_oversample(FLT0, 256);
```

### hpdf\_threshold\_monitor\_filter\_config

The description of hpdf\_threshold\_monitor\_filter\_config is shown as below:

**Table 3-600. Function hpdf\_threshold\_monitor\_filter\_config**

<b>Function name</b>	hpdf_threshold_monitor_filter_config
<b>Function prototype</b>	void hpdf_threshold_monitor_filter_config(hpdf_channel_enum channelx, uint32_t order, uint8_t oversample);
<b>Function descriptions</b>	configure threshold monitor filter order and oversample
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module

<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>order</b>	threshold monitor Sinc filter order
<i>TM_FASTSINC</i>	FastSinc filter type
<i>TM_SINC1</i>	Sinc1 filter type
<i>TM_SINC2</i>	Sinc2 filter type
<i>TM_SINC3</i>	Sinc3 filter type
<b>Input parameter{in}</b>	
<b>oversample</b>	Sinc filter oversampling rate(1-32)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure threshold monitor filter order and oversample */
```

```
hpdf_threshold_monitor_filter_config (FLT0, TM_SINC3);
```

### hpdf\_threshold\_monitor\_filter\_read\_data

The description of hpdf\_threshold\_monitor\_filter\_read\_data is shown as below:

**Table 3-601. Function hpdf\_threshold\_monitor\_filter\_read\_data**

<b>Function name</b>	hpdf_threshold_monitor_filter_read_data
<b>Function prototype</b>	int16_t hpdf_threshold_monitor_filter_read_data(hpdf_channel_enum channelx);
<b>Function descriptions</b>	read the threshold monitor filter data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int16_t</b>	the threshold monitor filter data

Example:

```
/* read the threshold monitor filter data */
```

```
int16_t data;
```

```
data = hpdf_threshold_monitor_filter_read_data(CHANNEL0);
```

## hpdf\_threshold\_monitor\_fast\_mode\_disable

The description of hpdf\_threshold\_monitor\_fast\_mode\_disable is shown as below:

**Table 3-602. Function hpdf\_threshold\_monitor\_fast\_mode\_disable**

<b>Function name</b>	hpdf_threshold_monitor_fast_mode_disable
<b>Function prototype</b>	void hpdf_threshold_monitor_fast_mode_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable threshold monitor fast mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable threshold monitor fast mode */
hpdf_threshold_monitor_fast_mode_disable (FLT0);
```

## hpdf\_threshold\_monitor\_fast\_mode\_enable

The description of hpdf\_threshold\_monitor\_fast\_mode\_enable is shown as below:

**Table 3-603. Function hpdf\_threshold\_monitor\_fast\_mode\_enable**

<b>Function name</b>	hpdf_threshold_monitor_fast_mode_enable
<b>Function prototype</b>	void hpdf_threshold_monitor_fast_mode_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable threshold monitor fast mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable threshold monitor fast mode */
hpdf_threshold_monitor_fast_mode_enable(FLT0);
```

## hpdf\_threshold\_monitor\_channel

The description of hpdf\_threshold\_monitor\_channel is shown as below:

**Table 3-604. Function hpdf\_threshold\_monitor\_channel**

<b>Function name</b>	hpdf_threshold_monitor_channel
<b>Function prototype</b>	void hpdf_threshold_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);
<b>Function descriptions</b>	configure threshold monitor channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>channel</b>	which channel use threshold monitor x
<i>TMCHEN_DISABLE</i>	extremes monitor x does not accept data from any channel
<i>TMCHEN_CHANNELx</i>	extremes monitor accepts data from channel x
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure threshold monitor channel */
```

```
hpdf_threshold_monitor_channel(FLT0, TMCHEN_CHANNEL1);
```

## hpdf\_threshold\_monitor\_high\_threshold

The description of hpdf\_threshold\_monitor\_high\_threshold is shown as below:

**Table 3-605. Function hpdf\_threshold\_monitor\_high\_threshold**

<b>Function name</b>	hpdf_threshold_monitor_high_threshold
<b>Function prototype</b>	void hpdf_threshold_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);
<b>Function descriptions</b>	configure threshold monitor high threshold value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>value</b>	high threshold value(-8388608~8388607)
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure threshold monitor high threshold value */
```

```
hpdf_threshold_monitor_high_threshold(FLT0, 32767);
```

### hpdf\_threshold\_monitor\_low\_threshold

The description of hpdf\_threshold\_monitor\_low\_threshold is shown as below:

**Table 3-606. Function hpdf\_threshold\_monitor\_low\_threshold**

<b>Function name</b>	hpdf_threshold_monitor_low_threshold
<b>Function prototype</b>	void hpdf_threshold_monitor_low_threshold(hpdf_filter_enum filtery, int32_t value);
<b>Function descriptions</b>	configure threshold monitor low threshold value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>value</b>	low threshold value(-8388608~8388607)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure threshold monitor low threshold value */
```

```
hpdf_threshold_monitor_low_threshold(FLT0, -32768);
```

### hpdf\_high\_threshold\_break\_signal

The description of hpdf\_high\_threshold\_break\_signal is shown as below:

**Table 3-607. Function hpdf\_high\_threshold\_break\_signal**

<b>Function name</b>	hpdf_high_threshold_break_signal
<b>Function prototype</b>	void hpdf_high_threshold_break_signal(hpdf_filter_enum filtery, uint32_t break_signal);
<b>Function descriptions</b>	configure threshold monitor high threshold event break signal
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
Input parameter{in}	
<b>break_signal</b>	HPDF break signal
<i>NO_TM_HT_BREAK</i>	break signal is not distributed to an threshold monitor high threshold event
<i>TM_HT_BREAKx</i> (x=0..3)	break signal x is distributed to an threshold monitor high threshold event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor high threshold event break signal */
```

```
hpdf_high_threshold_break_signal(FTL0, TM_HT_BREAK0);
```

### hpdf\_low\_threshold\_break\_signal

The description of hpdf\_low\_threshold\_break\_signal is shown as below:

**Table 3-608. Function hpdf\_low\_threshold\_break\_signal**

<b>Function name</b>	hpdf_low_threshold_break_signal
<b>Function prototype</b>	void hpdf_low_threshold_break_signal(hpdf_filter_enum filtery, uint32_t break_signal);
<b>Function descriptions</b>	configure threshold monitor low threshold event break signal
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
Input parameter{in}	
<b>break_signal</b>	HPDF break signal
<i>NO_TM_LT_BREAK</i>	break signal is not distributed to an threshold monitor low threshold event
<i>TM_LT_BREAKx</i> (x=0..3)	break signal x is distributed to an threshold monitor low threshold event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor low threshold event break signal */
```

```
hpdf_low_threshold_break_signal(FTL0, TM_LT_BREAK0);
```

## hpdf\_extremes\_monitor\_channel

The description of hpdf\_extremes\_monitor\_channel is shown as below:

**Table 3-609. Function hpdf\_extremes\_monitor\_channel**

<b>Function name</b>	hpdf_extremes_monitor_channel
<b>Function prototype</b>	void hpdf_extremes_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);
<b>Function descriptions</b>	configure extremes monitor channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>channel</b>	which channel use extremes monitor y (y=0..3)
<i>EM_CHANNEL_DISABLE</i>	extremes monitor y does not accept data from any channel
<i>EM_CHANNELx (x=0..7)</i>	extremes monitor accepts data from channel x
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure extremes monitor channel */
```

```
hpdf_extremes_monitor_channel(FTL0, EM_CHANNEL0);
```

## hpdf\_extremes\_monitor\_maximum\_get

The description of hpdf\_extremes\_monitor\_maximum\_get is shown as below:

**Table 3-610. Function hpdf\_extremes\_monitor\_maximum\_get**

<b>Function name</b>	hpdf_extremes_monitor_maximum_get
<b>Function prototype</b>	int32_t hpdf_extremes_monitor_maximum_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the extremes monitor maximum value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>

Output parameter{out}	
-	-
Return value	
int32_t	the maximum value

Example:

```
int32_t vlaue;
```

```
vlaue = hpdf_extremes_monitor_maximum_get(FTL0);
```

### hpdf\_extremes\_monitor\_minimum\_get

The description of hpdf\_extremes\_monitor\_minimum\_get is shown as below:

**Table 3-611. Function hpdf\_extremes\_monitor\_minimum\_get**

Function name	hpdf_extremes_monitor_minimum_get
Function prototype	int32_t hpdf_extremes_monitor_minimum_get(hpdf_filter_enum filtery);
Function descriptions	get the extremes monitor minimum value
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-
Return value	
int32_t	the minimum value

Example:

```
/* get the extremes monitor minimum value */
```

```
int32_t vlaue;
```

```
vlaue = hpdf_extremes_monitor_minimum_get(FTL0);
```

### hpdf\_conversion\_time\_get

The description of hpdf\_conversion\_time\_get is shown as below:

**Table 3-612. Function hpdf\_conversion\_time\_get**

Function name	hpdf_conversion_time_get
Function prototype	uint32_t hpdf_conversion_time_get(hpdf_filter_enum filtery);
Function descriptions	get the conversion timer value
Precondition	-
The called functions	-
Input parameter{in}	



<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int32_t</b>	the timer value

Example:

```
/* get the conversion timer value */
```

```
int32_t vlaue;
```

```
vlaue = hpdf_conversion_time_get(FTL0);
```

### hpdf\_rc\_continuous\_disable

The description of hpdf\_rc\_continuous\_disable is shown as below:

**Table 3-613. Function hpdf\_rc\_continuous\_disable**

<b>Function name</b>	hpdf_rc_continuous_disable
<b>Function prototype</b>	void hpdf_rc_continuous_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable regular conversions continuous mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable regular conversions continuous mode */
```

```
hpdf_rc_continuous_disable(FTL0);
```

### hpdf\_rc\_continuous\_enable

The description of hpdf\_rc\_continuous\_enable is shown as below:

**Table 3-614. Function hpdf\_rc\_continuous\_enable**

<b>Function name</b>	hpdf_rc_continuous_enable
<b>Function prototype</b>	void hpdf_rc_continuous_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable regular conversions continuous mode
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable regular conversions continuous mode */
```

```
hpdf_rc_continuous_enable(FTL0);
```

### hpdf\_rc\_start\_by\_software

The description of hpdf\_rc\_start\_by\_software is shown as below:

**Table 3-615. Function hpdf\_rc\_start\_by\_software**

Function name	hpdf_rc_start_by_software
Function prototype	void hpdf_rc_start_by_software(hpdf_filter_enum filtery);
Function descriptions	start regular channel conversion by software
Precondition	-
The called functions	-
Input parameter{in}	
filtery	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* start regular channel conversion by software */
```

```
hpdf_rc_start_by_software(FTL0);
```

### hpdf\_rc\_syn\_disable

The description of hpdf\_rc\_syn\_disable is shown as below:

**Table 3-616. Function hpdf\_rc\_syn\_disable**

Function name	hpdf_rc_syn_disable
Function prototype	void hpdf_rc_syn_disable(hpdf_filter_enum filtery);
Function descriptions	disable regular conversion synchronously

<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable regular conversion synchronously */
```

```
hpdf_rc_syn_disable(FTL0);
```

### hpdf\_rc\_syn\_enable

The description of hpdf\_rc\_syn\_enable is shown as below:

**Table 3-617. Function hpdf\_rc\_syn\_enable**

<b>Function name</b>	hpdf_rc_syn_enable
<b>Function prototype</b>	void hpdf_rc_syn_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable regular conversion synchronously
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable regular conversion synchronously */
```

```
hpdf_rc_syn_enable(FTL0);
```

### hpdf\_rc\_dma\_disable

The description of hpdf\_rc\_dma\_disable is shown as below:

**Table 3-618. Function hpdf\_rc\_dma\_disable**

<b>Function name</b>	hpdf_rc_dma_disable
<b>Function prototype</b>	void hpdf_rc_dma_disable(hpdf_filter_enum filtery);

<b>Function descriptions</b>	disable regular conversion DMA channel
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable regular conversion DMA channel */
hpdf_rc_dma_disable(FTL0);
```

### hpdf\_rc\_dma\_enable

The description of hpdf\_rc\_dma\_enable is shown as below:

**Table 3-619. Function hpdf\_rc\_dma\_enable**

<b>Function name</b>	hpdf_rc_dma_enable
<b>Function prototype</b>	void hpdf_rc_dma_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable regular conversion DMA channel
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable regular conversion DMA channel */
hpdf_rc_dma_enable(FTL0);
```

### hpdf\_rc\_channel\_config

The description of hpdf\_rc\_channel\_config is shown as below:

**Table 3-620. Function hpdf\_rc\_channel\_config**

<b>Function name</b>	hpdf_rc_channel_config
----------------------	------------------------

<b>Function prototype</b>	void hpdf_rc_channel_config(hpdf_filter_enum filtery, hpdf_channel_enum channelx);
<b>Function descriptions</b>	configure regular conversion channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-554. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure regular conversion channel */
hpdf_rc_channel_config(FTL0, CHANNEL1);
```

### hpdf\_rc\_fast\_mode\_disable

The description of hpdf\_rc\_fast\_mode\_disable is shown as below:

**Table 3-621. Function hpdf\_rc\_fast\_mode\_disable**

<b>Function name</b>	hpdf_rc_fast_mode_disable
<b>Function prototype</b>	void hpdf_rc_fast_mode_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable regular conversion fast conversion mode
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable regular conversion fast conversion mode */
hpdf_rc_fast_mode_disable(FTL0);
```

## hpdf\_rc\_fast\_mode\_enable

The description of hpdf\_rc\_fast\_mode\_enable is shown as below:

**Table 3-622. Function hpdf\_rc\_fast\_mode\_enable**

<b>Function name</b>	hpdf_rc_fast_mode_enable
<b>Function prototype</b>	void hpdf_rc_fast_mode_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable regular conversion fast conversion mode
<b>Precondition</b>	disable FLT <sub>y</sub> (y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLT<sub>y</sub></i> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable regular conversion fast conversion mode */
```

```
hpdf_rc_fast_mode_enable(FTL0);
```

## hpdf\_rc\_data\_get

The description of hpdf\_rc\_data\_get is shown as below:

**Table 3-623. Function hpdf\_rc\_data\_get**

<b>Function name</b>	hpdf_rc_data_get
<b>Function prototype</b>	int32_t hpdf_rc_data_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the regular conversion data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLT<sub>y</sub></i> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int32_t</b>	regular conversions data

Example:

```
/* get the regular conversion data */
```

```
int32_t vlaue;
```

```
vlaue = hpdf_rc_data_get(FTL0);
```

## hpdf\_rc\_channel\_get

The description of hpdf\_rc\_channel\_get is shown as below:

**Table 3-624. Function hpdf\_rc\_channel\_get**

<b>Function name</b>	hpdf_rc_channel_get
<b>Function prototype</b>	uint8_t hpdf_rc_channel_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the channel of regular channel most recently converted
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint8_t</b>	the channel

Example:

```
/* get the channel of regular channel most recently converted */
```

```
uint8_t channel;
```

```
channel = hpdf_rc_channel_get(FTL0);
```

## hpdf\_ic\_start\_by\_software

The description of hpdf\_ic\_start\_by\_software is shown as below:

**Table 3-625. Function hpdf\_ic\_start\_by\_software**

<b>Function name</b>	hpdf_ic_start_by_software
<b>Function prototype</b>	void hpdf_ic_start_by_software(hpdf_filter_enum filtery);
<b>Function descriptions</b>	start inserted channel conversion by software
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* start inserted channel conversion by software */
```

```
hpdf_ic_start_by_software(FTL0);
```

## hpdf\_ic\_syn\_disable

The description of hpdf\_ic\_syn\_disable is shown as below:

**Table 3-626. Function hpdf\_ic\_syn\_disable**

<b>Function name</b>	hpdf_ic_syn_disable
<b>Function prototype</b>	void hpdf_ic_syn_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable inserted conversion synchronously
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable inserted conversion synchronously */
```

```
hpdf_ic_syn_disable(FTL0);
```

## hpdf\_ic\_syn\_enable

The description of hpdf\_ic\_syn\_enable is shown as below:

**Table 3-627. Function hpdf\_ic\_syn\_enable**

<b>Function name</b>	hpdf_ic_syn_enable
<b>Function prototype</b>	void hpdf_ic_syn_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable inserted conversion synchronously
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:



```
/* enable inserted conversion synchronously */
```

```
hpdf_ic_syn_enable(FTL0);
```

## hpdf\_ic\_dma\_disable

The description of hpdf\_ic\_dma\_disable is shown as below:

**Table 3-628. Function hpdf\_ic\_dma\_disable**

<b>Function name</b>	hpdf_ic_dma_disable
<b>Function prototype</b>	void hpdf_ic_dma_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable inserted conversion DMA channel
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable inserted conversion DMA channel */
```

```
hpdf_ic_dma_disable(FTL0);
```

## hpdf\_ic\_dma\_enable

The description of hpdf\_ic\_dma\_enable is shown as below:

**Table 3-629. Function hpdf\_ic\_dma\_enable**

<b>Function name</b>	hpdf_ic_dma_enable
<b>Function prototype</b>	void hpdf_ic_dma_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable inserted conversion DMA channel
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable inserted conversion DMA channel */
```

```
hpdf_ic_dma_enable(FTL0);
```

## hpdf\_ic\_scan\_mode\_disable

The description of hpdf\_ic\_scan\_mode\_disable is shown as below:

**Table 3-630. Function hpdf\_ic\_scan\_mode\_disable**

<b>Function name</b>	hpdf_ic_scan_mode_disable
<b>Function prototype</b>	void hpdf_ic_scan_mode_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable scan conversion mode
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable scan conversion mode */
```

```
hpdf_ic_scan_mode_disable(FTL0);
```

## hpdf\_ic\_scan\_mode\_enable

The description of hpdf\_ic\_scan\_mode\_enable is shown as below:

**Table 3-631. Function hpdf\_ic\_scan\_mode\_enable**

<b>Function name</b>	hpdf_ic_scan_mode_enable
<b>Function prototype</b>	void hpdf_ic_scan_mode_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable scan conversion mode
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable scan conversion mode */
```

```
hpdf_ic_scan_mode_enable(FTL0);
```

## hpdf\_ic\_trigger\_signal\_disable

The description of hpdf\_ic\_trigger\_signal\_disable is shown as below:

**Table 3-632. Function hpdf\_ic\_trigger\_signal\_disable**

<b>Function name</b>	hpdf_ic_trigger_signal_disable
<b>Function prototype</b>	void hpdf_ic_trigger_signal_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable inserted conversions trigger signal
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable inserted conversions trigger signal */
```

```
hpdf_ic_trigger_signal_disable(FTL0);
```

## hpdf\_ic\_trigger\_signal\_config

The description of hpdf\_ic\_trigger\_signal\_config is shown as below:

**Table 3-633. Function hpdf\_ic\_trigger\_signal\_config**

<b>Function name</b>	hpdf_ic_trigger_signal_config
<b>Function prototype</b>	void hpdf_ic_trigger_signal_config(hpdf_filter_enum filtery, uint32_t trigger, uint32_t trigger_edge);
<b>Function descriptions</b>	configure inserted conversions trigger signal and trigger edge
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>trigger</b>	inserted conversions trigger signal
<i>HPDF_ITRG0</i>	TIMER0_TRGO0 is selected to start inserted conversion
<i>HPDF_ITRG1</i>	TIMER0_TRGO1 is selected to start inserted conversion

<i>HPDF_ITRG2</i>	TIMER7_TRGO0 is selected to start inserted conversion
<i>HPDF_ITRG3</i>	TIMER7_TRGO1 is selected to start inserted conversion
<i>HPDF_ITRG4</i>	TIMER2_TRGO0 is selected to start inserted conversion
<i>HPDF_ITRG5</i>	TIMER3_TRGO0 is selected to start inserted conversion
<i>HPDF_ITRG6</i>	TIMER15_CH1 is selected to start inserted conversion
<i>HPDF_ITRG7</i>	TIMER5_TRGO0 is selected to start inserted conversion
<i>HPDF_ITRG8</i>	TIMER6_TRGO0 is selected to start inserted conversion
<i>HPDF_ITRG24</i>	EXTI11 is selected to start inserted conversion
<i>HPDF_ITRG25</i>	EXTI15 is selected to start inserted conversion
<i>HPDF_ITRG31</i>	HPDF_ITRG is selected to start inserted conversion
<b>Input parameter{in}</b>	
<b>trigger_edge</b>	inserted conversions trigger edge
<i>TRG_DISABLE</i>	disable trigger signal
<i>RISING_EDGE_TRG</i>	rising edge on the trigger signal
<i>FALLING_EDGE_TRG</i>	falling edge on the trigger signal
<i>EDGE_TRG</i>	edge (rising edges and falling edges) on the trigger signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure inserted conversions trigger signal and trigger edge */
```

```
hpdf_ic_trigger_signal_config(FTL0, HPDF_ITRG3, FALLING_EDGE_TRG);
```

### hpdf\_ic\_channel\_config

The description of hpdf\_ic\_channel\_config is shown as below:

**Table 3-634. Function hpdf\_ic\_channel\_config**

<b>Function name</b>	hpdf_ic_channel_config
<b>Function prototype</b>	void hpdf_ic_channel_config(hpdf_filter_enum filtery, uint32_t channel);
<b>Function descriptions</b>	configure inserted group conversions channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>channel</b>	the HPDF channel belongs to inserted group
<i>IGCSEL_CHANNELx</i> (x=0..7)	Channel x belongs to the inserted group
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure inserted group conversions channel */
```

```
hpdf_ic_channel_config(FTL0, IGCSEL_CHANNEL1);
```

### hpdf\_ic\_data\_get

The description of hpdf\_ic\_data\_get is shown as below:

**Table 3-635. Function hpdf\_ic\_data\_get**

<b>Function name</b>	hpdf_ic_data_get
<b>Function prototype</b>	int32_t hpdf_ic_data_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the inserted conversions data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int32_t</b>	inserted conversions data

Example:

```
/* get the inserted conversions data */
```

```
int32_t vlaue;
```

```
vlaue = hpdf_ic_data_get(FTL0);
```

### hpdf\_ic\_channel\_get

The description of hpdf\_ic\_channel\_get is shown as below:

**Table 3-636. Function hpdf\_ic\_channel\_get**

<b>Function name</b>	hpdf_ic_channel_get
<b>Function prototype</b>	uint8_t hpdf_ic_channel_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the channel of inserted group channel most recently converted
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module

<i>FLTy</i> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint8_t</b>	the channel

Example:

```
/* get the channel of inserted group channel most recently converted */
```

```
uint8_t channel;
```

```
channel = hpdf_ic_channel_get(FTL0);
```

## hpdf\_flag\_get

The description of hpdf\_flag\_get is shown as below:

**Table 3-637. Function hpdf\_flag\_get**

<b>Function name</b>	hpdf_flag_get
<b>Function prototype</b>	FlagStatus hpdf_flag_get(hpdf_filter_enum filtery, hpdf_flag_enum flag);
<b>Function descriptions</b>	get the HPDF flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy</i> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	HPDF flags, refer to <a href="#">Table 3-556. Enum hpdf_flag_enum</a>
<i>HPDF_FLAG_FLTy_IC</i> <i>EF</i>	FLTy inserted conversion end flag
<i>HPDF_FLAG_FLTy_RC</i> <i>EF</i>	FLTy regular conversion end flag
<i>HPDF_FLAG_FLTy_IC</i> <i>DOF</i>	FLTy inserted conversion data overflow flag
<i>HPDF_FLAG_FLTy_RC</i> <i>DOF</i>	FLTy threshold monitor event occurred flag
<i>HPDF_FLAG_FLTy_TM</i> <i>EOF</i>	FLTy inserted conversion in progress flag
<i>HPDF_FLAG_FLTy_IC</i> <i>PF</i>	FLTy regular conversion in progress flag
<i>HPDF_FLAG_FLTy_RC</i> <i>PF</i>	clock signal is lost on channel 0 flag
<i>HPDF_FLAG_FLT0_CK</i> <i>LFx</i>	clock signal is lost on channel x flag
<i>HPDF_FLAG_FLT0_M</i>	malfunction event occurred on channel x flag

<i>MFx</i>	
<i>HPDF_FLAG_FLTy_RC</i> <i>HPDT</i>	FLTy inserted channel most recently converted
<i>HPDF_FLAG_FLTy_LT</i> <i>Fx</i>	threshold monitor low threshold flag on channel x flag
<i>HPDF_FLAG_FLTy_HT</i> <i>Fx</i>	threshold monitor high threshold flag on channel x flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the FLT0 threshold monitor event occurred flag */
```

```
hpdf_flag_get(FTL0, HPDF_FLAG_FLTy_TMEOF);
```

### hpdf\_flag\_clear

The description of hpdf\_flag\_clear is shown as below:

**Table 3-638. Function hpdf\_flag\_clear**

<b>Function name</b>	hpdf_flag_clear
<b>Function prototype</b>	void hpdf_flag_clear(hpdf_filter_enum filtery, hpdf_flag_enum flag);
<b>Function descriptions</b>	clear the HPDF flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy</i> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	HPDF flags, refer to <a href="#">Table 3-556. Enum hpdf_flag_enum</a>
<i>HPDF_FLAG_FLTy_IC</i> <i>EF</i>	FLTy inserted conversion end flag
<i>HPDF_FLAG_FLTy_RC</i> <i>EF</i>	FLTy regular conversion end flag
<i>HPDF_FLAG_FLTy_IC</i> <i>DOF</i>	FLTy inserted conversion data overflow flag
<i>HPDF_FLAG_FLTy_RC</i> <i>DOF</i>	FLTy threshold monitor event occurred flag
<i>HPDF_FLAG_FLTy_TM</i> <i>EOF</i>	FLTy inserted conversion in progress flag
<i>HPDF_FLAG_FLT0_CK</i> <i>LFx</i>	clock signal is lost on channel x flag

HPDF_FLAG_FLT0_M MFx	malfunction event occurred on channel x flag
HPDF_FLAG_FLTy_LT Fx	threshold monitor low threshold flag on channel x flag
HPDF_FLAG_FLTy_HT Fx	threshold monitor high threshold flag on channel x flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the FLT0 threshold monitor event occurred flag */
```

```
hpdf_flag_clear(FLT0, HPDF_FLAG_FLTy_TMEOF);
```

### hpdf\_interrupt\_enable

The description of hpdf\_interrupt\_enable is shown as below:

**Table 3-639. Function hpdf\_interrupt\_enable**

<b>Function name</b>	hpdf_interrupt_enable
<b>Function prototype</b>	void hpdf_interrupt_enable(hpdf_filter_enum filtery, hpdf_interrput_enum interrupt);
<b>Function descriptions</b>	enable HPDF interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
FLTy(y=0..3)	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
Input parameter{in}	
<b>interrupt</b>	HPDF interrupts, refer to <a href="#">Table 3-558. Enum hpdf_interrput_enum</a>
HPDF_INT_FLTy_ICEI E	FLTy inserted conversion end interrupt enable
HPDF_INT_FLTy_RCEI E	FLTy regular conversion end interrupt enable
HPDF_INT_FLTy_ICD OIE	FLTy inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_RCD OIE	FLTy regular conversion data overflow interrupt enable
HPDF_INT_FLTy_TMIE	FLTy threshold monitor interrupt enable
HPDF_INT_FLT0_MMI E	malfunction monitor interrupt enable
HPDF_INT_FLT0_CKLI	clock loss interrupt enable



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

Example:

```
/* enable FLT0 threshold monitor interrupt */
```

```
hpdf_interrupt_enable(FTL0, HPDF_INT_FLTy_TMEOIE);
```

### hpdf\_interrupt\_disable

The description of hpdf\_interrupt\_disable is shown as below:

**Table 3-640. Function hpdf\_interrupt\_disable**

<b>Function name</b>	hpdf_interrupt_disable
<b>Function prototype</b>	void hpdf_interrupt_disable(hpdf_filter_enum filtery, hpdf_interrput_enum interrupt);
<b>Function descriptions</b>	disable HPDF interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>interrupt</b>	HPDF interrupts, refer to <a href="#">Table 3-558. Enum hpdf_interrput_enum</a>
HPDF_INT_FLTy_ICEI E	FLTy inserted conversion end interrupt enable
HPDF_INT_FLTy_RCEI E	FLTy regular conversion end interrupt enable
HPDF_INT_FLTy_ICD OIE	FLTy inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_RCD OIE	FLTy regular conversion data overflow interrupt enable
HPDF_INT_FLTy_TMIE	FLTy threshold monitor interrupt enable
HPDF_INT_FLT0_MMI E	malfunction monitor interrupt enable
HPDF_INT_FLT0_CKLI E	clock loss interrupt enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FLT0 threshold monitor interrupt */
```

```
hpdf_interrupt_disable(FTL0, HPDF_INT_FLT0_CKLE);
```

## hpdf\_interrupt\_flag\_get

The description of hpdf\_interrupt\_flag\_get is shown as below:

**Table 3-641. Function hpdf\_interrupt\_flag\_get**

<b>Function name</b>	hpdf_interrupt_flag_get
<b>Function prototype</b>	FlagStatus hpdf_interrupt_flag_get(hpdf_filter_enum filtery, hpdf_interrput_flag_enum int_flag);
<b>Function descriptions</b>	get the HPDF interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>int_flag</b>	HPDF flags, refer to <a href="#">Table 3-557. Enum hpdf_interrput_flag_enum</a>
HPDF_INT_FLAG_FLT y_ICEF	FLTy inserted conversion end interrupt flag
HPDF_INT_FLAG_FLT y_RCEF	FLTy regular conversion end interrupt flag
HPDF_INT_FLAG_FLT y_ICDOF	FLTy inserted conversion data overflow interrupt flag
HPDF_INT_FLAG_FLT y_RCDOF	FLTy regular conversion data overflow interrupt flag
HPDF_INT_FLAG_FLT y_TMEOF	FLTy threshold monitor event occurred interrupt flag
HPDF_INT_FLAG_FLT 0_CKLFx	clock signal is lost on channel x interrupt flag
HPDF_INT_FLAG_FLT 0_MMFx	malfunction event occurred on channel x interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the the clock loss interrupt flag */
```

```
hpdf_interrupt_flag_get (FTL0, HPDF_INT_FLAG_FLT0_CKLF0);
```

## hpdf\_interrupt\_flag\_clear

The description of hpdf\_interrupt\_flag\_clear is shown as below:

**Table 3-642. Function hpdf\_interrupt\_flag\_clear**

<b>Function name</b>	hpdf_interrupt_flag_clear
<b>Function prototype</b>	void hpdf_interrupt_flag_clear(hpdf_filter_enum filtery, hpdf_interrput_flag_enum int_flag);
<b>Function descriptions</b>	clear the HPDF interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-555. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>int_flag</b>	HPDF flags, refer to <a href="#">Table 3-557. Enum hpdf_interrput_flag_enum</a>
HPDF_INT_FLAG_FLT y_ICEF	FLTy inserted conversion end interrupt flag
HPDF_INT_FLAG_FLT y_RCEF	FLTy regular conversion end interrupt flag
HPDF_INT_FLAG_FLT y_ICDOF	FLTy inserted conversion data overflow interrupt flag
HPDF_INT_FLAG_FLT y_RCDOF	FLTy regular conversion data overflow interrupt flag
HPDF_INT_FLAG_FLT y_TMEOF	FLTy threshold monitor event occurred interrupt flag
HPDF_INT_FLAG_FLT 0_CKLFx	clock signal is lost on channelx interrupt flag
HPDF_INT_FLAG_FLT 0_MMFx	malfunction event occurred on channelx interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the the clock loss interrupt flag */
hpdf_interrupt_flag_get(FTL0, HPDF_INT_FLAG_FLT0_CKLF0);
```

## 3.20. 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 [3.20.1](#), the I2C firmware functions are introduced in chapter [3.20.2](#).

### 3.20.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

**Table 3-643. I2C Registers**

Registers	Descriptions
I2C_CTL0	I2C control register 0
I2C_CTL1	I2C control register 1
I2C_SADDR0	I2C slave address register 0
I2C_SADDR1	I2C slave address register 1
I2C_TIMING	I2C timing register
I2C_TIMEOUT	I2C timeout register
I2C_STAT	I2C status register
I2C_STATC	I2C status clear register
I2C_PEC	I2C PEC register
I2C_RDATA	I2C receive data register
I2C_TDATA	I2C transmit data register
I2C_CTL2	I2C control register 2

### 3.20.2. Descriptions of Peripheral functions

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

**Table 3-644. 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 address and transfer direction in master mode
i2c_address10_header_enable	10-bit address header executes read direction only in master receive mode
i2c_address10_header_disable	10-bit address header executes complete sequence in master receive mode
i2c_address10_enable	enable 10-bit addressing mode in master mode
i2c_address10_disable	disable 10-bit addressing mode in master mode

Function name	Function description
i2c_automatic_end_enable	enable I2C automatic end mode in master mode
i2c_automatic_end_disable	disable I2C automatic end mode in master mode
i2c_slave_response_to_gcall_enable	enable the response to a general call
i2c_slave_response_to_gcall_disable	disable the response to a general call
i2c_stretch_scl_low_enable	enable to stretch SCL low when data is not ready in slave mode
i2c_stretch_scl_low_disable	disable to stretch SCL low when data is not ready in slave mode
i2c_address_config	configure i2c slave address
i2c_address_bit_compare_config	define which bits of ADDRESS[7:1] need to compare with the incoming address byte
i2c_address_disable	disable I2C address in slave mode
i2c_second_address_config	configure I2C second slave address
i2c_second_address_disable	disable I2C second address in slave mode
i2c_received_address_get	get received match address in slave mode
i2c_slave_byte_control_enable	enable slave byte control
i2c_slave_byte_control_disable	disable slave byte control
i2c_nack_enable	generate a NACK in slave mode
i2c_nack_disable	generate an ACK in slave mode
i2c_wakeup_from_deepsleep_enable	enable wakeup from deep-sleep mode
i2c_wakeup_from_deepsleep_disable	disable wakeup from deep-sleep mode
i2c_enable	enable I2C
i2c_disable	disable I2C
i2c_start_on_bus	generate a START condition on I2C bus
i2c_stop_on_bus	generate a STOP condition on I2C bus
i2c_data_transmit	I2C transmit data
i2c_data_receive	I2C receive data
i2c_reload_enable	enable I2C reload mode
i2c_reload_disable	disable I2C reload mode
i2c_transfer_byte_number_config	configure number of bytes to be transferred
i2c_dma_enable	enable I2C DMA for transmission or reception
i2c_dma_disable	disable I2C DMA for transmission or reception
i2c_pec_transfer	I2C transfers PEC value
i2c_pec_enable	enable I2C PEC calculation
i2c_pec_disable	disable I2C PEC calculation
i2c_pec_value_get	get packet error checking value
i2c_smbus_alert_enable	enable SMBus alert
i2c_smbus_alert_disable	disable SMBus alert
i2c_smbus_default_addr_enable	enable SMBus device default address
i2c_smbus_default_addr_disable	disable SMBus device default address
i2c_smbus_host_addr_enable	enable SMBus host address

Function name	Function description
i2c_smbus_host_addr_disable	disable SMBus host address
i2c_extented_clock_timeout_enable	enable extended clock timeout detection
i2c_extented_clock_timeout_disable	disable extended clock timeout detection
i2c_clock_timeout_enable	enable clock timeout detection
i2c_clock_timeout_disable	disable clock timeout detection
i2c_bus_timeout_b_config	configure bus timeout B
i2c_bus_timeout_a_config	configure bus timeout A
i2c_idle_clock_timeout_config	configure idle clock timeout detection
i2c_flag_get	get I2C flag status
i2c_flag_clear	clear I2C flag status
i2c_interrupt_enable	enable I2C interrupt
i2c_interrupt_disable	disable I2C interrupt
i2c_interrupt_flag_get	get I2C interrupt flag status
i2c_interrupt_flag_clear	clear I2C interrupt flag status

### Enum i2c\_interrupt\_flag\_enum

Table 3-645. Enum i2c\_interrupt\_flag\_enum

Member name	Function description
I2C_INT_FLAG_TI	transmit interrupt flag
I2C_INT_FLAG_RBNE	I2C_RDATA is not empty during receiving interrupt flag
I2C_INT_FLAG_ADDSEND	address received matches in slave mode interrupt flag
I2C_INT_FLAG_NACK	not acknowledge interrupt flag
I2C_INT_FLAG_STPDET	stop condition detected in slave mode interrupt flag
I2C_INT_FLAG_TC	transfer complete in master mode interrupt flag
I2C_INT_FLAG_TCR	transfer complete reload interrupt flag
I2C_INT_FLAG_BERR	bus error interrupt flag
I2C_INT_FLAG_LOSTARB	arbitration lost interrupt flag
I2C_INT_FLAG_OUERR	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 3-646. 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}	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
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 3-647. Function i2c\_timing\_config**

<b>Function name</b>	i2c_timing_config
<b>Function prototype</b>	void i2c_timing_config(uint32_t i2c_periph, uint32_t psc, uint32_t scl_dely, uint32_t sda_dely);
<b>Function descriptions</b>	configure the timing parameters
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
<b>psc</b>	0-0xf, timing prescaler
Input parameter{in}	
<b>scl_dely</b>	0-0xf, data setup time
Input parameter{in}	
<b>sda_dely</b>	0-0xf, data hold time
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the timing parameters */
i2c_timing_config(I2C0, 0x1, 0x2, 0x1);
```

## i2c\_digital\_noise\_filter\_config

The description of i2c\_digital\_noise\_filter\_config is shown as below:

**Table 3-648. Function i2c\_digital\_noise\_filter\_config**

<b>Function name</b>	i2c_digital_noise_filter_config
<b>Function prototype</b>	void i2c_digital_noise_filter_config(uint32_t i2c_periph, uint32_t filter_length);
<b>Function descriptions</b>	configure digital noise filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>filter_length</b>	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 <sub>I2CCLK</sub>
<i>FILTER_LENGTH_2</i>	digital filter is enabled and filter spikes with a length of up to 2 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_3</i>	digital filter is enabled and filter spikes with a length of up to 3 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_4</i>	digital filter is enabled and filter spikes with a length of up to 4 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_5</i>	digital filter is enabled and filter spikes with a length of up to 5 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_6</i>	digital filter is enabled and filter spikes with a length of up to 6 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_7</i>	digital filter is enabled and filter spikes with a length of up to 7 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_8</i>	digital filter is enabled and filter spikes with a length of up to 8 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_9</i>	digital filter is enabled and filter spikes with a length of up to 9 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_10</i>	digital filter is enabled and filter spikes with a length of up to 10 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_11</i>	digital filter is enabled and filter spikes with a length of up to 11 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_12</i>	digital filter is enabled and filter spikes with a length of up to 12 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_13</i>	digital filter is enabled and filter spikes with a length of up to 13 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_14</i>	digital filter is enabled and filter spikes with a length of up to 14 t <sub>I2CCLK</sub>
<i>FILTER_LENGTH_15</i>	digital filter is enabled and filter spikes with a length of up to 15 t <sub>I2CCLK</sub>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* I2C0 digital filter filters spikes with a length of up to 1 tI2CCLK */
```

```
i2c_digital_noise_filter_config(I2C0, FILTER_LENGTH_1);
```



## i2c\_analog\_noise\_filter\_enable

The description of i2c\_analog\_noise\_filter\_enable is shown as below:

**Table 3-649. Function i2c\_analog\_noise\_filter\_enable**

<b>Function name</b>	i2c_analog_noise_filter_enable
<b>Function prototype</b>	void i2c_analog_noise_filter_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable analog noise filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-650. Function i2c\_analog\_noise\_filter\_disable**

<b>Function name</b>	i2c_analog_noise_filter_disable
<b>Function prototype</b>	void i2c_analog_noise_filter_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable analog noise filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-651. Function i2c\_master\_clock\_config**

<b>Function name</b>	i2c_master_clock_config
<b>Function prototype</b>	void i2c_master_clock_config(uint32_t i2c_periph, uint32_t sclh, uint32_t scll);
<b>Function descriptions</b>	configure the SCL high and low period of clock in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>sclh</b>	0-0xff, SCL high period
<b>Input parameter{in}</b>	
<b>scll</b>	0-0xff, SCL low period
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-652. Function i2c\_master\_addressing**

<b>Function name</b>	i2c_master_addressing
<b>Function prototype</b>	void i2c_master_addressing(uint32_t i2c_periph, uint32_t address, uint32_t trans_direction);
<b>Function descriptions</b>	configure i2c slave addresss and transfer direction in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>address</b>	0-0x3FF except reserved address, I2C slave address to be sent
<b>Input parameter{in}</b>	

<b>trans_direction</b>	I2C transfer direction in master mode
<i>I2C_MASTER_TRANSMIT</i>	master transmit
<i>I2C_MASTER_RECEIVE</i>	master receive
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-653. Function i2c\_address10\_header\_enable**

<b>Function name</b>	i2c_address10_header_enable
<b>Function prototype</b>	void i2c_address10_header_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	10-bit address header executes read direction only in master receive mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-654. Function i2c\_address10\_header\_disable**

<b>Function name</b>	i2c_address10_header_disable
<b>Function prototype</b>	void i2c_address10_header_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	10-bit address header executes complete sequence in master receive mode

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-655. Function i2c\_address10\_enable**

<b>Function name</b>	i2c_address10_enable
<b>Function prototype</b>	void i2c_address10_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable 10-bit addressing mode in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-656. Function i2c\_address10\_disable**

<b>Function name</b>	i2c_address10_disable
<b>Function prototype</b>	void i2c_address10_disable(uint32_t i2c_periph);

<b>Function descriptions</b>	disable 10-bit addressing mode in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-657. Function i2c\_automatic\_end\_enable**

<b>Function name</b>	i2c_automatic_end_enable
<b>Function prototype</b>	void i2c_automatic_end_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable I2C automatic end mode in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-658. Function i2c\_automatic\_end\_disable**

<b>Function name</b>	i2c_automatic_end_disable
----------------------	---------------------------

<b>Function prototype</b>	void i2c_automatic_end_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable I2C automatic end mode in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-659. Function i2c\_slave\_response\_to\_gcall\_enable**

<b>Function name</b>	i2c_slave_response_to_gcall_enable
<b>Function prototype</b>	void i2c_slave_response_to_gcall_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable the response to a general call
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-660. Function `i2c_slave_response_to_gcall_disable`

<b>Function name</b>	<code>i2c_slave_response_to_gcall_disable</code>
<b>Function prototype</b>	<code>void i2c_slave_response_to_gcall_disable(uint32_t i2c_periph);</code>
<b>Function descriptions</b>	disable the response to a general call
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-661. Function `i2c_stretch_scl_low_enable`

<b>Function name</b>	<code>i2c_stretch_scl_low_enable</code>
<b>Function prototype</b>	<code>void i2c_stretch_scl_low_enable(uint32_t i2c_periph);</code>
<b>Function descriptions</b>	enable to stretch SCL low when data is not ready in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-662. Function i2c\_stretch\_scl\_low\_disable

<b>Function name</b>	i2c_stretch_scl_low_disable
<b>Function prototype</b>	void i2c_stretch_scl_low_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable to stretch SCL low when data is not ready in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-663. Function i2c\_address\_config

<b>Function name</b>	i2c_address_config
<b>Function prototype</b>	void i2c_address_config(uint32_t i2c_periph, uint32_t address, uint32_t addr_format);
<b>Function descriptions</b>	configure i2c slave address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>address</b>	I2C address
<b>Input parameter{in}</b>	
<b>addr_format</b>	7 bits or 10 bits
<i>I2C_ADDFORMAT_7BITS</i>	7bits
<i>I2C_ADDFORMAT_10BITS</i>	10bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	



-	-
---	---

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 3-664. 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
I2Cx	(x=0,1,2,3)
Input parameter{in}	
compare_bits	the bits need to compare
ADDRESS_BIT1_COMPARE	address bit1 needs compare
ADDRESS_BIT2_COMPARE	address bit2 needs compare
ADDRESS_BIT3_COMPARE	address bit3 needs compare
ADDRESS_BIT4_COMPARE	address bit4 needs compare
ADDRESS_BIT5_COMPARE	address bit5 needs compare
ADDRESS_BIT6_COMPARE	address bit6 needs compare
ADDRESS_BIT7_COMPARE	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 3-665. Function i2c\_address\_disable**

<b>Function name</b>	i2c_address_disable
<b>Function prototype</b>	void i2c_address_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable i2c address in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-666. Function i2c\_second\_address\_config**

<b>Function name</b>	i2c_second_address_config
<b>Function prototype</b>	void i2c_second_address_config(uint32_t i2c_periph, uint32_t address, uint32_t addr_mask);
<b>Function descriptions</b>	configure i2c second slave address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>address</b>	I2C address
<b>Input parameter{in}</b>	
<b>addr_mask</b>	the bits not need to compare

<code>ADDRESS2_NO_MAS K</code>	no mask, all the bits must be compared
<code>ADDRESS2_MASK_BIT1</code>	ADDRESS2[1] is masked, only ADDRESS2[7:2] are compared
<code>ADDRESS2_MASK_BIT1_2</code>	ADDRESS2[2:1] is masked, only ADDRESS2[7:3] are compared
<code>ADDRESS2_MASK_BIT1_3</code>	ADDRESS2[3:1] is masked, only ADDRESS2[7:4] are compared
<code>ADDRESS2_MASK_BIT1_4</code>	ADDRESS2[4:1] is masked, only ADDRESS2[7:5] are compared
<code>ADDRESS2_MASK_BIT1_5</code>	ADDRESS2[5:1] is masked, only ADDRESS2[7:6] are compared
<code>ADDRESS2_MASK_BIT1_6</code>	ADDRESS2[6:1] is masked, only ADDRESS2[7] are compared
<code>ADDRESS2_MASK_ALL L</code>	all the ADDRESS2[7:1] bits are masked
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-667. Function i2c\_second\_address\_disable**

<b>Function name</b>	i2c_second_address_disable
<b>Function prototype</b>	void i2c_second_address_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable i2c second address in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-668. Function i2c\_receved\_address\_get**

<b>Function name</b>	i2c_receved_address_get
<b>Function prototype</b>	uint32_t i2c_receved_address_get(uint32_t i2c_periph);
<b>Function descriptions</b>	get received match address in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	0x00..0x7F

Example:

```
/* get received match address in slave mode */
```

```
uint32_t address;
```

```
address = i2c_receved_address_get(I2C0);
```

### i2c\_slave\_byte\_control\_enable

The description of i2c\_slave\_byte\_control\_enable is shown as below:

**Table 3-669. Function i2c\_slave\_byte\_control\_enable**

<b>Function name</b>	i2c_slave_byte_control_enable
<b>Function prototype</b>	void i2c_slave_byte_control_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable slave byte control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-670. Function i2c\_slave\_byte\_control\_disable**

<b>Function name</b>	i2c_slave_byte_control_disable
<b>Function prototype</b>	void i2c_slave_byte_control_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable slave byte control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-671. Function i2c\_nack\_enable**

<b>Function name</b>	i2c_nack_enable
<b>Function prototype</b>	void i2c_nack_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	generate a NACK in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-672. Function i2c\_nack\_disable**

<b>Function name</b>	i2c_nack_disable
<b>Function prototype</b>	void i2c_nack_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	generate a ACK in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* generate a ACK in slave mode */
```

```
i2c_nack_disable(I2C0);
```

### **i2c\_wakeup\_from\_deepsleep\_enable**

The description of i2c\_wakeup\_from\_deepsleep\_enable is shown as below:

**Table 3-673. Function i2c\_wakeup\_from\_deepsleep\_enable**

<b>Function name</b>	i2c_wakeup_from_deepsleep_enable
<b>Function prototype</b>	void i2c_wakeup_from_deepsleep_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable wakeup from Deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable wakeup from Deep-sleep mode */

i2c_wakeup_from_deepsleep_enable(I2C0);
```

### i2c\_wakeup\_from\_deepsleep\_disable

The description of i2c\_wakeup\_from\_deepsleep\_disable is shown as below:

**Table 3-674. Function i2c\_wakeup\_from\_deepsleep\_disable**

<b>Function name</b>	i2c_wakeup_from_deepsleep_disable
<b>Function prototype</b>	void i2c_wakeup_from_deepsleep_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable wakeup from Deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable wakeup from Deep-sleep mode */

i2c_wakeup_from_deepsleep_disable(I2C0);
```

### i2c\_enable

The description of i2c\_enable is shown as below:

**Table 3-675. Function i2c\_enable**

<b>Function name</b>	i2c_enable
<b>Function prototype</b>	void i2c_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable I2C
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I2C0 */

i2c_enable(I2C0);
```

### i2c\_disable

The description of i2c\_disable is shown as below:

**Table 3-676. Function i2c\_disable**

<b>Function name</b>	i2c_disable
<b>Function prototype</b>	void i2c_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable I2C
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable I2C0 */

i2c_disable(I2C0);
```

### i2c\_start\_on\_bus

The description of i2c\_start\_on\_bus is shown as below:

**Table 3-677. Function i2c\_start\_on\_bus**

<b>Function name</b>	i2c_start_on_bus
<b>Function prototype</b>	void i2c_start_on_bus(uint32_t i2c_periph);
<b>Function descriptions</b>	generate a START condition on I2C bus
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-



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 3-678. Function i2c\_stop\_on\_bus**

<b>Function name</b>	i2c_stop_on_bus
<b>Function prototype</b>	void i2c_stop_on_bus(uint32_t i2c_periph);
<b>Function descriptions</b>	generate a STOP condition on I2C bus
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-679. Function i2c\_data\_transmit**

<b>Function name</b>	i2c_data_transmit
<b>Function prototype</b>	void i2c_data_transmit(uint32_t i2c_periph, uint32_t data);
<b>Function descriptions</b>	I2C transmit data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>data</b>	transmit data
<b>Output parameter{out}</b>	
-	-

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 3-680. 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
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
uint32_t	received data (0x00-0xFF)

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 3-681. 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
I2Cx	(x=0,1,2,3)

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 3-682. 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
I2Cx	(x=0,1,2,3)
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 3-683. 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}	

<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>byte_number</b>	0x0-0xFF, number of bytes to be transferred
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-684. Function i2c\_dma\_enable**

<b>Function name</b>	i2c_dma_enable
<b>Function prototype</b>	void i2c_dma_enable(uint32_t i2c_periph, uint8_t dma);
<b>Function descriptions</b>	enable I2C DMA for transmission or reception
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>dma</b>	I2C DMA
<i>I2C_DMA_TRANSMIT</i>	transmit data using DMA
<i>I2C_DMA_RECEIVE</i>	receive data using DMA
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-685. 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
I2Cx	(x=0,1,2,3)
Input parameter{in}	
dma	I2C DMA
I2C_DMA_TRANSMIT	transmit data using DMA
I2C_DMA_RECEIVE	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 3-686. 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=0,1,2,3)
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 3-687. Function i2c\_pec\_enable**

<b>Function name</b>	i2c_pec_enable
<b>Function prototype</b>	void i2c_pec_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable I2C PEC calculation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I2C PEC calculation */
```

```
i2c_pec_enable(I2C0);
```

## i2c\_pec\_disable

The description of i2c\_pec\_disable is shown as below:

**Table 3-688. Function i2c\_pec\_disable**

<b>Function name</b>	i2c_pec_disable
<b>Function prototype</b>	void i2c_pec_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable I2C PEC calculation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-689. Function i2c\_pec\_value\_get**

<b>Function name</b>	i2c_pec_value_get
<b>Function prototype</b>	uint32_t i2c_pec_value_get(uint32_t i2c_periph);
<b>Function descriptions</b>	get packet error checking value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	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 3-690. Function i2c\_smbus\_alert\_enable**

<b>Function name</b>	i2c_smbus_alert_enable
<b>Function prototype</b>	void i2c_smbus_alert_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable SMBus Alert
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-691. Function i2c\_smbus\_alert\_disable**

<b>Function name</b>	i2c_smbus_alert_disable
<b>Function prototype</b>	void i2c_smbus_alert_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable SMBus Alert
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-692. Function i2c\_smbus\_default\_addr\_enable**

<b>Function name</b>	i2c_smbus_default_addr_enable
<b>Function prototype</b>	void i2c_smbus_default_addr_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable SMBus device default address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-693. Function i2c\_smbus\_default\_addr\_disable**

<b>Function name</b>	i2c_smbus_default_addr_disable
<b>Function prototype</b>	void i2c_smbus_default_addr_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable SMBus device default address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-694. Function i2c\_smbus\_host\_addr\_enable**

<b>Function name</b>	i2c_smbus_host_addr_enable
<b>Function prototype</b>	void i2c_smbus_host_addr_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable SMBus Host address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-695. Function i2c\_smbus\_host\_addr\_disable**

<b>Function name</b>	i2c_smbus_host_addr_disable
<b>Function prototype</b>	void i2c_smbus_host_addr_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable SMBus Host address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-696. Function i2c\_extented\_clock\_timeout\_enable**

<b>Function name</b>	i2c_extented_clock_timeout_enable
<b>Function prototype</b>	void i2c_extented_clock_timeout_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable extended clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-697. Function i2c\_extented\_clock\_timeout\_disable**

<b>Function name</b>	i2c_extented_clock_timeout_disable
<b>Function prototype</b>	void i2c_extented_clock_timeout_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable extended clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-698. Function i2c\_clock\_timeout\_enable**

<b>Function name</b>	i2c_clock_timeout_enable
<b>Function prototype</b>	void i2c_clock_timeout_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-699. Function i2c\_clock\_timeout\_disable**

<b>Function name</b>	i2c_clock_timeout_disable
<b>Function prototype</b>	void i2c_clock_timeout_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-700. Function i2c\_bus\_timeout\_b\_config**

<b>Function name</b>	i2c_bus_timeout_b_config
<b>Function prototype</b>	void i2c_bus_timeout_b_config(uint32_t i2c_periph, uint32_t timeout);
<b>Function descriptions</b>	configure bus timeout B
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>timeout</b>	0-0xffff, bus timeout B
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-701. Function i2c\_bus\_timeout\_a\_config**

<b>Function name</b>	i2c_bus_timeout_a_config
<b>Function prototype</b>	void i2c_bus_timeout_a_config(uint32_t i2c_periph, uint32_t timeout);
<b>Function descriptions</b>	configure bus timeout A
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>timeout</b>	0-0xffff, bus timeout A
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-702. Function i2c\_idle\_clock\_timeout\_config**

<b>Function name</b>	i2c_idle_clock_timeout_config
<b>Function prototype</b>	void i2c_idle_clock_timeout_config(uint32_t i2c_periph, uint32_t timeout);
<b>Function descriptions</b>	configure idle clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>timeout</b>	bus timeout A
<i>BUSTOA_DETECT_SC</i>	BUSTOA is used to detect SCL low timeout

<i>L_LOW</i>	
<i>BUSTOA_DETECT_ID</i> <i>LE</i>	BUSTOA is used to detect both SCL and SDA high timeout when the bus is idle
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-703. Function i2c\_flag\_get**

<b>Function name</b>	i2c_flag_get
<b>Function prototype</b>	FlagStatus i2c_flag_get(uint32_t i2c_periph, uint32_t flag);
<b>Function descriptions</b>	get I2C flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>flag</b>	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>	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
<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 3-704. Function i2c\_flag\_clear**

Function name	i2c_flag_clear
Function prototype	void i2c_flag_clear(uint32_t i2c_periph, uint32_t flag);
Function descriptions	clear I2C flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
flag	I2C flags
I2C_FLAG_ADDSEND	address received matches in slave mode
I2C_FLAG_NACK	not acknowledge flag
I2C_FLAG_STPDET	STOP condition detected in slave mode
I2C_FLAG_BERR	bus error
I2C_FLAG_LOSTARB	arbitration Lost
I2C_FLAG_OUERR	overrun/underrun error in slave mode
I2C_FLAG_PECERR	PEC error
I2C_FLAG_TIMEOUT	timeout flag
I2C_FLAG_SMBALT	SMBus Alert
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear a bus error flag*/
```

```
i2c_flag_clear(I2C0, I2C_FLAG_BERR);
```

## i2c\_interrupt\_enable

The description of i2c\_interrupt\_enable is shown as below:

**Table 3-705. Function i2c\_interrupt\_enable**

<b>Function name</b>	i2c_interrupt_enable
<b>Function prototype</b>	void i2c_interrupt_enable(uint32_t i2c_periph, uint32_t interrupt);
<b>Function descriptions</b>	enable I2C interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>interrupt</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-706. Function i2c\_interrupt\_disable**

<b>Function name</b>	i2c_interrupt_disable
<b>Function prototype</b>	void i2c_interrupt_disable(uint32_t i2c_periph, uint32_t interrupt);
<b>Function descriptions</b>	disable I2C interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)



Input parameter{in}	
<b>interrupt</b>	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 3-707. Function i2c\_interrupt\_flag\_get**

<b>Function name</b>	i2c_interrupt_flag_get
<b>Function prototype</b>	FlagStatus i2c_interrupt_flag_get(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	get I2C interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
<b>int_flag</b>	I2C interrupt flags, refer to <a href="#">Table 3-645. Enum i2c_interrupt_flag_enum.</a>
<i>I2C_INT_FLAG_TI</i>	transmit interrupt flag
<i>I2C_INT_FLAG_RBNE</i>	I2C_RDATA is not empty during receiving interrupt flag
<i>I2C_INT_FLAG_ADDS 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_TC</i>	transfer complete in master mode interrupt flag
<i>I2C_INT_FLAG_TCR</i>	transfer complete reload interrupt flag

<i>I2C_INT_FLAG_BERR</i>	bus error interrupt flag
<i>I2C_INT_FLAG_LOSTARB</i>	arbitration lost interrupt flag
<i>I2C_INT_FLAG_OUERR</i>	overflow/underrun error in slave mode interrupt flag
<i>I2C_INT_FLAG_PECERR</i>	PEC error interrupt flag
<i>I2C_INT_FLAG_TIMEOUT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBALERT</i>	SMBus Alert interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-708. Function i2c\_interrupt\_flag\_clear**

<b>Function name</b>	i2c_interrupt_flag_clear
<b>Function prototype</b>	void i2c_interrupt_flag_clear(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	clear I2C interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>int_flag</b>	I2C interrupt flags, refer to <a href="#">Table 3-645. Enum i2c_interrupt_flag_enum</a> .
<i>I2C_INT_FLAG_ADDS</i> <i>END</i>	address received matches in slave mode interrupt flag
<i>I2C_INT_FLAG_NACK</i>	not acknowledge interrupt flag
<i>I2C_INT_FLAG_STPDET</i> <i>ET</i>	stop condition detected in slave mode interrupt flag
<i>I2C_INT_FLAG_BERR</i>	bus error interrupt flag

<i>I2C_INT_FLAG_LOSTA</i> <i>RB</i>	arbitration lost interrupt flag
<i>I2C_INT_FLAG_OUER</i> <i>R</i>	overflow/underrun error in slave mode interrupt flag
<i>I2C_INT_FLAG_PEC</i> <i>RR</i>	PEC error interrupt flag
<i>I2C_INT_FLAG_TIMEO</i> <i>UT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBA</i> <i>LT</i>	SMBus Alert interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear a bus error flag */
```

```
i2c_interrupt_flag_clear(I2C0, I2C_INT_FLAG_BERR);
```

## 3.21. LPTIMER

The LPTIMER is a 16-bit / 32-bit timer and it is able to keep running in all power modes except for Standby mode with its diversity of clock sources. The LPTIMER registers are listed in chapter [3.21.1](#), the LPTIMER firmware functions are introduced in chapter [3.21.2](#).

### 3.21.1. Descriptions of Peripheral registers

LPTIMER registers are listed in the table shown as below:

**Table 3-709. LPTIMER Registers**

Registers	Descriptions
LPTIMER_INTF	interrupt flag register
LPTIMER_INTC	interrupt flag clear register
LPTIMER_INTEN	interrupt enable register
LPTIMER_CTL0	control register 0
LPTIMER_CTL1	control register 1
LPTIMER_CMPV	compare value register
LPTIMER_CAR	counter auto reload register
LPTIMER_CNT	counter register
LPTIMER_EIRMP	external input remap register
LPTIMER_INHLCMV	input high level counter max value register

### 3.21.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

**Table 3-710. LPTIMER firmware function**

Function name	Function description
lptimer_deinit	deinit the LPTIMER
lptimer_struct_para_init	initialize LPTIMER init parameter struct with a default value
lptimer_init	initialize LPTIMER counter
lptimer_inputremap	configure external input remap
lptimer_register_shadow_enable	enable the LPTIMER_CAR and LPTIMER_CMPV registers shadow function
lptimer_register_shadow_disable	disable the LPTIMER_CAR and LPTIMER_CMPV registers shadow function
lptimer_timeout_enable	enable the LPTIMER TIMEOUT function
lptimer_timeout_disable	disable the LPTIMER TIMEOUT function
lptimer_counter_sync_reset	counter synchronously reset
lptimer_counter_read_async_reset_enable	read cause counter asynchronously reset enable
lptimer_counter_read_async_reset_disable	read cause counter asynchronously reset disable
lptimer_continue_start	LPTIMER start with countinue mode
lptimer_single_start	LPTIMER start with single mode
lptimer_stop	stop LPTIMER
lptimer_counter_read	read LPTIMER current counter value
lptimer_autoreload_read	read LPTIMER auto reload value
lptimer_compare_read	read LPTIMER compare value
lptimer_autoreload_value_config	configure LPTIMER autoreload register value
lptimer_compare_value_config	configure LPTIMER compare value
lptimer_decodemode0_enable	enable decode mode 0
lptimer_decodemode1_enable	enable decode mode 1
lptimer_decodemode_disable	disable decode mode 0/1
lptimer_highlevelcounter_enable	enable external input high level counter
lptimer_highlevelcounter_disable	disable external input high level counter
lptimer_flag_get	get LPTIMER flags
lptimer_flag_clear	clear LPTIMER flags
lptimer_interrupt_enable	enable the LPTIMER interrupt
lptimer_interrupt_disable	disable the LPTIMER interrupt
lptimer_interrupt_flag_get	get LPTIMER interrupt flag
lptimer_interrupt_flag_clear	clear LPTIMER interrupt flag

## Structure `lptimer_parameter_struct`

**Table 3-711. Structure `lptimer_parameter_struct`**

Member name	Function description
<code>clocksource</code>	clock source (LPTIMER_INTERNALCLK, LPTIMER_EXTERNALCLK)
<code>prescaler</code>	counter clock prescaler (LPTIMER_PSC_x, x=1,2,4,8..128)
<code>extclockpolarity</code>	external clock polarity of the active edge for the counter (LPTIMER_EXTERNALCLK_RISING, LPTIMER_EXTERNALCLK_FALLING, LPTIMER_EXTERNALCLK_BOTH)
<code>extclockfilter</code>	external clock sampling time to configure the clock glitch filter (LPTIMER_EXTERNALCLK_FILTEROFF, LPTIMER_EXTERNALCLK_FILTER_2, LPTIMER_EXTERNALCLK_FILTER_4, LPTIMER_EXTERNALCLK_FILTER_8)
<code>triggermode</code>	trigger mode (LPTIMER_TRIGGER_SOFTWARE, LPTIMER_TRIGGER_EXTERNALRISING, LPTIMER_TRIGGER_EXTERNALFALLING, LPTIMER_TRIGGER_EXTERNALBOTH)
<code>extriggersource</code>	external trigger source (LPTIMER_EXTRIGGER_GPIO, LPTIMER_EXTRIGGER_RTCALARM0, LPTIMER_EXTRIGGER_RTCALARM1, LPTIMER_EXTRIGGER_RTCTAMP0, LPTIMER_EXTRIGGER_RTCTAMP1, LPTIMER_EXTRIGGER_RTCTAMP2, LPTIMER_EXTRIGGER_CMP0_OUT, LPTIMER_EXTRIGGER_CMP1_OUT, LPTIMER_EXTRIGGER_CMP2_OUT, LPTIMER_EXTRIGGER_CMP3_OUT, LPTIMER_EXTRIGGER_CMP4_OUT, LPTIMER_EXTRIGGER_CMP5_OUT, LPTIMER_EXTRIGGER_CMP6_OUT, LPTIMER_EXTRIGGER_CMP7_OUT)
<code>extriggerfilter</code>	external trigger filter (LPTIMER_TRIGGER_FILTEROFF, LPTIMER_TRIGGER_FILTER_2, LPTIMER_TRIGGER_FILTER_4, LPTIMER_TRIGGER_FILTER_8)
<code>outputpolarity</code>	output polarity (LPTIMER_OUTPUT_NOTINVERTED, LPTIMER_OUTPUT_INVERTED)
<code>outputmode</code>	output mode (LPTIMER_OUTPUT_PWMORSINGLE, LPTIMER_OUTPUT_SET)
<code>countersource</code>	counter source (LPTIMER_COUNTER_INTERNAL, LPTIMER_COUNTER_EXTERNAL)

## `lptimer_deinit`

The description of `lptimer_deinit` is shown as below:

**Table 3-712. Function `lptimer_deinit`**

<b>Function name</b>	<code>lptimer_deinit</code>
<b>Function prototype</b>	<code>void lptimer_deinit(void);</code>
<b>Function descriptions</b>	deinit LPTIMER

<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinit LPTIMER */
```

```
lptimer_deinit();
```

### **lptimer\_struct\_para\_init**

The description of lptimer\_struct\_para\_init is shown as below:

**Table 3-713. Function lptimer\_struct\_para\_init**

<b>Function name</b>	lptimer_struct_para_init
<b>Function prototype</b>	void lptimer_struct_para_init(lptimer_parameter_struct *initpara);
<b>Function descriptions</b>	initialize LPTIMER init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>initpara</b>	LPTIMER init parameter struct, the structure members can refer to <a href="#">Table 3-711. Structure lptimer_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize LPTIMER init parameter struct with a default value */
```

```
lptimer_parameter_struct lptimer_initpara;
```

```
lptimer_struct_para_init(&lptimer_initpara);
```

### **lptimer\_init**

The description of lptimer\_init is shown as below:

**Table 3-714. Function lptimer\_init**

<b>Function name</b>	lptimer_init
<b>Function prototype</b>	void lptimer_init(lptimer_parameter_struct *initpara);

<b>Function descriptions</b>	initialize LPTIMER counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Input parameter{in}</b>	
initpara	TIMER init parameter struct, the structure members can refer to <a href="#">Table 3-711. Structure lptimer_parameter_struct.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize LPTIMER */
```

```
lptimer_parameter_struct lptimer_structure;
```

```
lptimer_structure.clocksource      = LPTIMER_INTERNALCLK;
```

```
lptimer_structure.prescaler       = LPTIMER_PSC_16;
```

```
lptimer_structure.extclockpolarity = LPTIMER_EXTERNALCLK_RISING;
```

```
lptimer_structure.extclockfilter   = LPTIMER_EXTERNALCLK_FILTEROFF;
```

```
lptimer_structure.triggermode      = LPTIMER_TRIGGER_SOFTWARE;
```

```
lptimer_structure.extriggersource  = LPTIMER_EXTRIGGER_GPIO;
```

```
lptimer_structure.extriggerfilter  = LPTIMER_TRIGGER_FILTEROFF;
```

```
lptimer_structure.outputpolarity  = LPTIMER_OUTPUT_NOTINVERTED;
```

```
lptimer_structure.outputmode       = LPTIMER_OUTPUT_PWMORSINGLE;
```

```
lptimer_structure.countersource    = LPTIMER_COUNTER_INTERNAL;
```

```
lptimer_init(lptimer_structure);
```

## lptimer\_inputremap

The description of lptimer\_inputremap is shown as below:

**Table 3-715. Function lptimer\_inputremap**

<b>Function name</b>	lptimer_inputremap
<b>Function prototype</b>	void lptimer_inputremap(uint32_t input0remap, uint32_t input1remap);
<b>Function descriptions</b>	configure external input remap
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>input0remap</b>	external input0 remap
<i>LPTIMER_INPUT0_GPIO</i>	external input is remaped to GPIO
<i>LPTIMER_INPUT0_CMP0_OUT</i>	external input is remaped to CMP0_OUT
<i>LPTIMER_INPUT0_CMP2_OUT</i>	external input is remaped to CMP2_OUT
<i>LPTIMER_INPUT0_CMP4_OUT</i>	external input is remaped to CMP4_OUT
<i>LPTIMER_INPUT0_CMP6_OUT</i>	external input is remaped to CMP6_OUT
Input parameter{in}	
<b>input1remap</b>	external input1 remap
<i>LPTIMER_INPUT1_GPIO</i>	external input is remaped to GPIO
<i>LPTIMER_INPUT1_CMP1_OUT</i>	external input is remaped to CMP1_OUT
<i>LPTIMER_INPUT1_CMP3_OUT</i>	external input is remaped to CMP3_OUT
<i>LPTIMER_INPUT1_CMP5_OUT</i>	external input is remaped to CMP5_OUT
<i>LPTIMER_INPUT1_CMP7_OUT</i>	external input is remaped to CMP7_OUT
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure LPTIMER inputs is connected to GPIO */
```

```
lptimer_inputremap(LPTIMER_INPUT0_GPIO, LPTIMER_INPUT1_GPIO);
```

### **lptimer\_register\_shadow\_enable**

The description of lptimer\_register\_shadow\_enable is shown as below:

**Table 3-716. Function lptimer\_register\_shadow\_enable**

<b>Function name</b>	lptimer_register_shadow_enable
<b>Function prototype</b>	void lptimer_register_shadow_enable(void);
<b>Function descriptions</b>	enable the LPTIMER_CAR and LPTIMER_CMPV registers shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-



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

Example:

```
/* enable the LPTIMER_CAR and LPTIMER_CMPV registers shadow function */
```

```
lptimer_register_shadow_enable();
```

### lptimer\_register\_shadow\_disable

The description of lptimer\_register\_shadow\_disable is shown as below:

**Table 3-717. Function lptimer\_register\_shadow\_disable**

Function name	lptimer_register_shadow_disable
Function prototype	void lptimer_register_shadow_disable(void);
Function descriptions	disable the LPTIMER_CAR and LPTIMER_CMPV registers shadow function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the LPTIMER_CAR and LPTIMER_CMPV registers shadow function */
```

```
lptimer_register_shadow_disable();
```

### lptimer\_timeout\_enable

The description of lptimer\_timeout\_enable is shown as below:

**Table 3-718. Function lptimer\_timeout\_enable**

Function name	lptimer_timeout_enable
Function prototype	void lptimer_timeout_enable(void);
Function descriptions	enable the LPTIMER TIMEOUT function
Precondition	-
The called functions	-

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

Example:

```
/* enable the LPTIMER TIMEOUT function */
```

```
lptimer_timeout_enable();
```

### **lptimer\_timeout\_disable**

The description of lptimer\_timeout\_disable is shown as below:

**Table 3-719. Function lptimer\_timeout\_disable**

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

Example:

```
/* disable the LPTIMER TIMEOUT function */
```

```
lptimer_timeout_disable();
```

### **lptimer\_counter\_sync\_reset**

The description of lptimer\_counter\_sync\_reset is shown as below:

**Table 3-720. Function lptimer\_timeout\_enable**

Function name	lptimer_counter_sync_reset
Function prototype	void lptimer_counter_sync_reset(void);
Function descriptions	LPTIMER counter reset synchronously
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* LPTIMER counter reset synchronously */
```

```
lptimer_counter_sync_reset();
```

### **lptimer\_counter\_read\_async\_reset\_enable**

The description of lptimer\_counter\_read\_async\_reset\_enable is shown as below:

**Table 3-721. Function lptimer\_timeout\_enable**

<b>Function name</b>	lptimer_counter_read_async_reset_enable
<b>Function prototype</b>	void lptimer_counter_read_async_reset_enable(void);
<b>Function descriptions</b>	enable read cause counter asynchronously reset
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read cause counter asynchronously reset enable */
```

```
lptimer_counter_read_async_reset_enable();
```

### **lptimer\_counter\_read\_async\_reset\_disable**

The description of lptimer\_counter\_read\_async\_reset\_disable is shown as below:

**Table 3-722. Function lptimer\_timeout\_enable**

<b>Function name</b>	lptimer_counter_read_async_reset_disable
<b>Function prototype</b>	void lptimer_counter_read_async_reset_disable(void);
<b>Function descriptions</b>	disable read cause counter asynchronously reset
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read cause counter asynchronously reset disable */
```

```
lptimer_counter_read_async_reset_disable();
```

### **lptimer\_countinue\_start**

The description of lptimer\_countinue\_start is shown as below:

**Table 3-723. Function lptimer\_countinue\_start**

Function name	lptimer_countinue_start
Function prototype	void lptimer_countinue_start(uint32_t autoreload, uint32_t compare);
Function descriptions	LPTIMER countinue start
Precondition	-
The called functions	-
Input parameter{in}	
autoreload	auto reload value, 0x0~0xFFFF
Input parameter{in}	
compare	0x0~0xFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* LPTIMER countinue start */
```

```
lptimer_countinue_start(0x0000FFFF, 0x00007FFF);
```

### **lptimer\_single\_start**

The description of lptimer\_single\_start is shown as below:

**Table 3-724. Function lptimer\_single\_start**

Function name	lptimer_single_start
Function prototype	void lptimer_single_start(uint32_t autoreload, uint32_t compare);
Function descriptions	LPTIMER single start
Precondition	-
The called functions	-
Input parameter{in}	

<b>autoreload</b>	auto reload value, 0x0~0xFFFF
<b>Input parameter{in}</b>	
<b>compare</b>	0x0~0xFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* LPTIMER single start */
```

```
lptimer_single_start(0x0000FFFF, 0x00007FFF);
```

### lptimer\_stop

The description of lptimer\_stop is shown as below:

**Table 3-725. Function lptimer\_stop**

<b>Function name</b>	lptimer_stop
<b>Function prototype</b>	void lptimer_stop(void);
<b>Function descriptions</b>	stop LPTIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* stop LPTIMER */
```

```
lptimer_stop();
```

### lptimer\_counter\_read

The description of lptimer\_counter\_read is shown as below:

**Table 3-726. Function lptimer\_counter\_read**

<b>Function name</b>	lptimer_counter_read
<b>Function prototype</b>	uint32_t lptimer_counter_read(void);
<b>Function descriptions</b>	read LPTIMER current counter value
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit counter value (0x0~0xFFFF)

Example:

```
/* read LPTIMER current counter value */
```

```
uint32_t i = 0;
```

```
i = lptimer_counter_read();
```

### lptimer\_autoreload\_read

The description of lptimer\_autoreload\_read is shown as below:

**Table 3-727. Function lptimer\_autoreload\_read**

Function name	lptimer_autoreload_read
Function prototype	uint32_t lptimer_autoreload_read(void);
Function descriptions	read LPTIMER auto reload value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit auto reload value (0x0~0xFFFF)

Example:

```
/* read LPTIMER auto reload value */
```

```
uint32_t i = 0;
```

```
i = lptimer_autoreload_read();
```

### lptimer\_compare\_read

The description of lptimer\_compare\_read is shown as below:

**Table 3-728. Function lptimer\_compare\_read**

Function name	lptimer_compare_read
Function prototype	uint32_t lptimer_compare_read(void);
Function descriptions	read LPTIMER compare value
Precondition	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit compare value (0x0~0xFFFF)

Example:

```
/* read LPTIMER compare value */
uint32_t i = 0;
i = lptimer_compare_read();
```

### lptimer\_autoreload\_value\_config

The description of lptimer\_autoreload\_value\_config is shown as below:

**Table 3-729. Function lptimer\_autoreload\_value\_config**

Function name	lptimer_autoreload_value_config
Function prototype	void lptimer_autoreload_value_config(uint32_t autoreload);
Function descriptions	configure LPTIMER autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
autoreload	autoreload value (0x0~0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure LPTIMER autoreload register value */
lptimer_autoreload_value_config(0x00FF);
```

### lptimer\_compare\_value\_config

The description of lptimer\_compare\_value\_config is shown as below:

**Table 3-730. Function lptimer\_compare\_value\_config**

Function name	lptimer_compare_value_config
Function prototype	void lptimer_compare_value_config(uint32_t compare);
Function descriptions	configure LPTIMER compare value
Precondition	-

The called functions	-
Input parameter{in}	
counter	compare value (0x0~0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure LPTIMER compare value */
lptimer_compare_value_config(0x00FF);
```

### **lptimer\_decodemode0\_enable**

The description of lptimer\_decodemode0\_enable is shown as below:

**Table 3-731. Function lptimer\_decodemode0\_enable**

Function name	lptimer_decodemode0_enable
Function prototype	void lptimer_decodemode0_enable(void);
Function descriptions	enable decode mode 0
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable decode mode 0 */
lptimer_decodemode0_enable();
```

### **lptimer\_decodemode1\_enable**

The description of lptimer\_decodemode1\_enable is shown as below:

**Table 3-732. Function lptimer\_decodemode1\_enable**

Function name	lptimer_decodemode1_enable
Function prototype	void lptimer_decodemode1_enable(void);
Function descriptions	enable decode mode 1
Precondition	-
The called functions	-



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

Example:

```
/* enable decode mode 1 */
```

```
lptimer_decodemode1_enable();
```

### **lptimer\_decodemode\_disable**

The description of lptimer\_decodemode\_disable is shown as below:

**Table 3-733. Function lptimer\_decodemode\_disable**

Function name	lptimer_decodemode_disable
Function prototype	void lptimer_decodemode_disable(void);
Function descriptions	disable decode mode 0/1
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable decode mode 0/1 */
```

```
lptimer_decodemode_disable();
```

### **lptimer\_highlevelcounter\_enable**

The description of lptimer\_highlevelcounter\_enable is shown as below:

**Table 3-734. Function lptimer\_highlevelcounter\_enable**

Function name	lptimer_highlevelcounter_enable
Function prototype	void lptimer_highlevelcounter_enable(uint32_t maxvalue);
Function descriptions	enable external input high level counter
Precondition	-
The called functions	-
Input parameter{in}	

-	-
<b>Input parameter{in}</b>	
<b>maxvalue</b>	input high level counter max value, 0x0~0x03FFFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable external input high level counter */
lptimer_highlevelcounter_enable(0x00007FFF);
```

### **lptimer\_highlevelcounter\_disable**

The description of lptimer\_highlevelcounter\_disable is shown as below:

**Table 3-735. Function lptimer\_highlevelcounter\_disable**

<b>Function name</b>	lptimer_highlevelcounter_disable
<b>Function prototype</b>	void lptimer_highlevelcounter_disable(void);
<b>Function descriptions</b>	disable external input high level counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable external input high level counter */
lptimer_highlevelcounter_disable();
```

### **lptimer\_flag\_get**

The description of lptimer\_flag\_get is shown as below:

**Table 3-736. Function lptimer\_flag\_get**

<b>Function name</b>	lptimer_flag_get
<b>Function prototype</b>	FlagStatus lptimer_flag_get(uint32_t flag);
<b>Function descriptions</b>	get LPTIMER flags
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>flag</b>	the LPTIMER flag
<i>LPTIMER_FLAG_CMPVM</i>	compare value register match flag
<i>LPTIMER_FLAG_CARM</i>	counter auto reload register match flag
<i>LPTIMER_FLAG_ETED</i>	external trigger edge event flag
<i>LPTIMER_FLAG_CMPVUP</i>	compare value register update flag
<i>LPTIMER_FLAG_CARUP</i>	counter auto reload register update flag
<i>LPTIMER_FLAG_UP</i>	LPTIMER counter direction change down to up flag
<i>LPTIMER_FLAG_DOWN</i>	LPTIMER counter direction change up to down flag
<i>LPTIMER_FLAG_HLCMVUP</i>	input high level counter max value register update flag
<i>LPTIMER_FLAG_INHLO</i>	LPTIMER_INx(x=0,1) high level counter overflow flag
<i>LPTIMER_FLAG_INHLOE</i>	the high level of LPTIMER_IN0 and LPTIMER_IN1 overlap error flag
<i>LPTIMER_FLAG_INRFOE</i>	the falling and rising edges of LPTIMER_IN0 and LPTIMER_IN1 overlap error flag
<i>LPTIMER_FLAG_IN0E</i>	LPTIMER_IN0 error flag
<i>LPTIMER_FLAG_IN1E</i>	LPTIMER_IN1 error flag
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get LPTIMER flag */
```

```
FlagStatus Flag_status = RESET;
```

```
Flag_status = lptimer_flag_get(LPTIMER_FLAG_CMPVM);
```

### lptimer\_flag\_clear

The description of lptimer\_flag\_clear is shown as below:

**Table 3-737. Function lptimer\_flag\_clear**

<b>Function name</b>	lptimer_flag_clear
<b>Function prototype</b>	void lptimer_flag_clear(uint32_t flag);
<b>Function descriptions</b>	clear LPTIMER flags
<b>Precondition</b>	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	the LPTIMER flag
<i>LPTIMER_FLAG_CMPVM</i>	compare value register match flag
<i>LPTIMER_FLAG_CARM</i>	counter auto reload register match flag
<i>LPTIMER_FLAG_ETED</i>	external trigger edge event flag
<i>LPTIMER_FLAG_CMPVUP</i>	compare value register update flag
<i>LPTIMER_FLAG_CARUP</i>	counter auto reload register update flag
<i>LPTIMER_FLAG_UP</i>	LPTIMER counter direction change down to up flag
<i>LPTIMER_FLAG_DOW</i>	LPTIMER counter direction change up to down flag
<i>LPTIMER_FLAG_HLCMVUP</i>	input high level counter max value register update flag
<i>LPTIMER_FLAG_INHLC</i>	LPTIMER_INx(x=0,1) high level counter overflow flag
<i>LPTIMER_FLAG_INHLO</i>	the high level of LPTIMER_IN0 and LPTIMER_IN1 overlap error flag
<i>LPTIMER_FLAG_INRFOE</i>	the falling and rising edges of LPTIMER_IN0 and LPTIMER_IN1 overlap error flag
<i>LPTIMER_FLAG_IN0E</i>	LPTIMER_IN0 error flag
<i>LPTIMER_FLAG_IN1E</i>	LPTIMER_IN1 error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear LPTIMER flag */
```

```
lptimer_flag_clear(LPTIMER_FLAG_CMPVM);
```

### **lptimer\_interrupt\_enable**

The description of lptimer\_interrupt\_enable is shown as below:

**Table 3-738. Function lptimer\_interrupt\_enable**

<b>Function name</b>	lptimer_interrupt_enable
<b>Function prototype</b>	void lptimer_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable the LPTIMER interrupt
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
interrupt	LPTIMER interrupt source
LPTIMER_INT_CMPVM	compare value register match interrupt
LPTIMER_INT_CARM	counter auto reload register match interrupt
LPTIMER_INT_ETEDEV	external trigger edge event interrupt
LPTIMER_INT_CMPVUP	compare value register update interrupt
LPTIMER_INT_CARUP	counter auto reload register update interrupt
LPTIMER_INT_UP	LPTIMER counter direction change down to up interrupt
LPTIMER_INT_DOWN	LPTIMER counter direction change up to down interrupt
LPTIMER_INT_HLCMVUP	input high level counter max value register update interrupt
LPTIMER_INT_INHOCO	LPTIMER_INx(x=0,1) high level counter overflow interrupt
LPTIMER_INT_INHLOE	the high level of LPTIMER_IN0 and LPTIMER_IN1 overlap error interrupt
LPTIMER_INT_INRFOE	the falling and rising edges of LPTIMER_IN0 and LPTIMER_IN1 overlap error interrupt
LPTIMER_INT_IN0E	LPTIMER_IN0 error interrupt
LPTIMER_INT_IN1E	LPTIMER_IN1 error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the LPTIMER interrupt */
```

```
lptimer_interrupt_enable(LPTIMER_INT_CMPVM);
```

### lptimer\_interrupt\_disable

The description of lptimer\_interrupt\_disable is shown as below:

**Table 3-739. Function lptimer\_interrupt\_disable**

Function name	lptimer_interrupt_disable
Function prototype	void lptimer_interrupt_disable(uint32_t interrupt);
Function descriptions	disable the LPTIMER interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	LPTIMER interrupt source
LPTIMER_INT_CMPVM	compare value register match interrupt
LPTIMER_INT_CARM	counter auto reload register match interrupt
LPTIMER_INT_ETEDEV	external trigger edge event interrupt

<i>LPTIMER_INT_CMPVUP</i>	compare value register update interrupt
<i>LPTIMER_INT_CARUP</i>	counter auto reload register update interrupt
<i>LPTIMER_INT_UP</i>	LPTIMER counter direction change down to up interrupt
<i>LPTIMER_INT_DOWN</i>	LPTIMER counter direction change up to down interrupt
<i>LPTIMER_INT_HLCMVUP</i>	input high level counter max value register update interrupt
<i>LPTIMER_INT_INHLCO</i>	LPTIMER_INx(x=0,1) high level counter overflow interrupt
<i>LPTIMER_INT_INHLOE</i>	the high level of LPTIMER_IN0 and LPTIMER_IN1 overlap error interrupt
<i>LPTIMER_INT_INRFOE</i>	the falling and rising edges of LPTIMER_IN0 and LPTIMER_IN1 overlap error interrupt
<i>LPTIMER_INT_IN0E</i>	LPTIMER_IN0 error interrupt
<i>LPTIMER_INT_IN1E</i>	LPTIMER_IN1 error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the LPTIMER interrupt */
```

```
lptimer_interrupt_disable(LPTIMER_INT_CMPVM);
```

### lptimer\_interrupt\_flag\_get

The description of lptimer\_interrupt\_flag\_get is shown as below:

**Table 3-740. Function lptimer\_interrupt\_flag\_get**

<b>Function name</b>	lptimer_interrupt_flag_get
<b>Function prototype</b>	FlagStatus lptimer_interrupt_flag_get(uint32_t int_flag);
<b>Function descriptions</b>	get LPTIMER interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	the LPTIMER interrupt flag
<i>LPTIMER_INT_FLAG_CMPVM</i>	compare value register match interrupt flag
<i>LPTIMER_INT_FLAG_CARM</i>	counter auto reload register match interrupt flag
<i>LPTIMER_INT_FLAG_ETEDEV</i>	external trigger edge event interrupt flag
<i>LPTIMER_INT_FLAG_CMPVUP</i>	compare value register update interrupt flag
<i>LPTIMER_INT_FLAG_CCARUP</i>	counter auto reload register update interrupt flag

<i>ARUP</i>	
<i>LPTIMER_INT_FLAG_UP</i>	LPTIMER counter direction change down to up interrupt flag
<i>LPTIMER_INT_FLAG_DOWN</i>	LPTIMER counter direction change up to down interrupt flag
<i>LPTIMER_INT_FLAG_HLCMVUP</i>	input high level counter max value register update interrupt flag
<i>LPTIMER_INT_FLAG_IHNLCO</i>	LPTIMER_INx(x=0,1) high level counter overflow interrupt flag
<i>LPTIMER_INT_FLAG_IHNL0E</i>	the high level of LPTIMER_IN0 and LPTIMER_IN1 overlap error interrupt flag
<i>LPTIMER_INT_FLAG_INRFOE</i>	the falling and rising edges of LPTIMER_IN0 and LPTIMER_IN1 overlap error interrupt flag
<i>LPTIMER_INT_FLAG_IN0E</i>	LPTIMER_IN0 error interrupt flag
<i>LPTIMER_INT_FLAG_IN1E</i>	LPTIMER_IN1 error interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get LPTIMER interrupt flag */
```

```
FlagStatus Flag_interrupt = RESET;
```

```
Flag_interrupt = lptimer_interrupt_flag_get(LPTIMER_INT_FLAG_CMPVM);
```

### **lptimer\_interrupt\_flag\_clear**

The description of lptimer\_interrupt\_flag\_clear is shown as below:

**Table 3-741. Function lptimer\_interrupt\_flag\_clear**

<b>Function name</b>	lptimer_interrupt_flag_clear
<b>Function prototype</b>	void lptimer_interrupt_flag_clear(uint32_t int_flag);
<b>Function descriptions</b>	clear LPTIMER interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	the LPTIMER interrupt flag
<i>LPTIMER_INT_FLAG_CMPVM</i>	compare value register match interrupt flag
<i>LPTIMER_INT_FLAG_CARM</i>	counter auto reload register match interrupt flag

<i>LPTIMER_INT_FLAG_ETEDEV</i>	external trigger edge event interrupt flag
<i>LPTIMER_INT_FLAG_CMPVUP</i>	compare value register update interrupt flag
<i>LPTIMER_INT_FLAG_CARRUP</i>	counter auto reload register update interrupt flag
<i>LPTIMER_INT_FLAG_UP</i>	LPTIMER counter direction change down to up interrupt flag
<i>LPTIMER_INT_FLAG_DOWN</i>	LPTIMER counter direction change up to down interrupt flag
<i>LPTIMER_INT_FLAG_HLCMVUP</i>	input high level counter max value register update interrupt flag
<i>LPTIMER_INT_FLAG_I_NHLCO</i>	LPTIMER_INx(x=0,1) high level counter overflow interrupt flag
<i>LPTIMER_INT_FLAG_I_NHLOE</i>	the high level of LPTIMER_IN0 and LPTIMER_IN1 overlap error interrupt flag
<i>LPTIMER_INT_FLAG_I_NRFOE</i>	the falling and rising edges of LPTIMER_IN0 and LPTIMER_IN1 overlap error interrupt flag
<i>LPTIMER_INT_FLAG_I_N0E</i>	LPTIMER_IN0 error interrupt flag
<i>LPTIMER_INT_FLAG_I_N1E</i>	LPTIMER_IN1 error interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear LPTIMER interrupt flag */
```

```
lptimer_interrupt_flag_clear(LPTIMER_INT_FLAG_CMPVM);
```

## 3.22. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick. The NVIC and SysTick registers are listed in chapter [3.22.1](#), the MISC firmware functions are introduced in chapter [3.22.2](#).

### 3.22.1. Descriptions of Peripheral registers

**Table 3-742. NVIC Registers**

Registers	Descriptions
ISER <sup>(1)</sup>	Interrupt Set Enable Register



Registers	Descriptions
ICER <sup>(1)</sup>	Interrupt Clear Enable Register
ISPR <sup>(1)</sup>	Interrupt Set Pending Register
ICPR <sup>(1)</sup>	Interrupt Clear Pending Register
IABR <sup>(1)</sup>	Interrupt Active bit Register
ITNS <sup>(1)</sup>	Interrupt Non-Secure State Register
IPR <sup>(1)</sup>	Interrupt Priority Register (8Bit wide)
STIR <sup>(1)</sup>	Software Trigger Interrupt Register
CPUID <sup>(2)</sup>	CPUID Base Register
ICSR <sup>(2)</sup>	Interrupt Control and State Register
VTOR <sup>(2)</sup>	Vector Table Offset Register
AIRCR <sup>(2)</sup>	Application Interrupt and Reset Control Register
SCR <sup>(2)</sup>	System Control Register
CCR <sup>(2)</sup>	Configuration Control Register
SHPR <sup>(2)</sup>	System Handlers Priority Registers
SHCSR <sup>(2)</sup>	System Handler Control and State Register
CFSR <sup>(2)</sup>	Configurable Fault Status Register
HFSR <sup>(2)</sup>	HardFault Status Register
DFSR <sup>(2)</sup>	Debug Fault Status Register
MMFAR <sup>(2)</sup>	MemManage Fault Address Register
BFAR <sup>(2)</sup>	BusFault Address Register
AFSR <sup>(2)</sup>	Auxiliary Fault Status Register
ID_PFR <sup>(2)</sup>	Processor Feature Register
ID_DFR <sup>(2)</sup>	Debug Feature Register
ID_AFR <sup>(2)</sup>	Auxiliary Feature Register
ID_MFR <sup>(2)</sup>	Memory Model Feature Register
ID_ISAR <sup>(2)</sup>	Instruction Set Attributes Register
CLIDR <sup>(2)</sup>	Cache Level ID register
CTR <sup>(2)</sup>	Cache Type register
CCSIDR <sup>(2)</sup>	Cache Size ID Register
CSSELR <sup>(2)</sup>	Cache Size Selection Register
CPACR <sup>(2)</sup>	Coprocessor Access Control Register
NSACR <sup>(2)</sup>	Non-Secure Access Control Register
SFSR <sup>(2)</sup>	Secure Fault Status Register
SFAR <sup>(2)</sup>	Secure Fault Address Register
STIR <sup>(2)</sup>	Software Triggered Interrupt Register
MVFR0 <sup>(2)</sup>	Media and VFP Feature Register 0
MVFR1 <sup>(2)</sup>	Media and VFP Feature Register 1
MVFR2 <sup>(2)</sup>	Media and VFP Feature Register 2
ICIALLU <sup>(2)</sup>	I-Cache Invalidate All to PoU
ICIMVAU <sup>(2)</sup>	I-Cache Invalidate by MVA to PoU
DCIMVAC <sup>(2)</sup>	D-Cache Invalidate by MVA to PoC

Registers	Descriptions
DCISW <sup>(2)</sup>	D-Cache Invalidate by Set-way
DCCMVAU <sup>(2)</sup>	D-Cache Clean by MVA to PoU
DCCMVAC <sup>(2)</sup>	D-Cache Clean by MVA to PoC
DCCSW <sup>(2)</sup>	D-Cache Clean by Set-way
DCCIMVAC <sup>(2)</sup>	D-Cache Clean and Invalidate by MVA to PoC
DCCISW <sup>(2)</sup>	D-Cache Clean and Invalidate by Set-way
BPIALL <sup>(2)</sup>	Branch Predictor Invalidate All

1. refer to the structure NVIC\_Type, is defined in the core\_cm33.h file

2. refer to the structure SCB\_Type, is defined in the core\_cm33.h file

**Table 3-743. SysTick Registers**

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

1. refer to the structure SysTick\_Type, is defined in the core\_cm33.h file

### 3.22.2. Descriptions of Peripheral functions

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

**Table 3-744. MISC firmware function**

Function name	Function description
nvic_priority_group_set	set the priority group
nvic_irq_enable	enable NVIC request
nvic_irq_disable	disable NVIC request
nvic_system_reset	initiates a system reset request to reset the MCU
nvic_vector_table_set	set the NVIC vector table address
system_lowpower_set	set the state of the low power mode
system_lowpower_reset	reset the state of the low power mode
systick_clksource_set	set the systick clock source
mpu_enable	enable the MPU
mpu_disable	disable the MPU
mpu_region_struct_para_init	initialize mpu_region_init_struct with the default values
mpu_attribute_struct_para_init	initialize mpu_attribute_init_struct with the default values
mpu_region_config	configure the MPU region
mpu_attribute_config	configure the MPU attribute
mpu_region_enable	enable MPU region

## Structure mpu\_region\_init\_struct

**Table 3-745. Structure mpu\_region\_init\_struct**

Member name	Function description
region_number	region number
region_base_address	region base address
region_limit_address	region limit address
access_permission	access permissions(AP) field
instruction_exec	execute never
shareability	defines the shareability for Normal memory
attribute_index	attribute index

## Structure mpu\_attribute\_init\_struct

**Table 3-746. Structure mpu\_attribute\_init\_struct**

Member name	Function description
attribute_number	attribute number
memory_type	memory type
outer_attributes	outer memory attributes
inner_attributes	inner memory attributes

## Enum IRQn\_Type

**Table 3-747. Enum IRQn\_Type**

Member name	Function description
WWDGT_IRQn	WWDGT interrupt
LVD_VAVD_VOVD_VUVD_IRQn	LVD / VAVD / VOVD / VUVD through EXTI line detect interrupt
TAMPER_IRQn	tamper through EXTI line detect
RTC_IRQn	RTC through EXTI line interrupt
FMC_IRQn	FMC global interrupt
RCU_IRQn	RCU global interrupt
EXTI0_IRQn	EXTI Line 0 interrupt
EXTI1_IRQn	EXTI Line 1 interrupt
EXTI2_IRQn	EXTI Line 2 interrupt
EXTI3_IRQn	EXTI Line 3 interrupt
EXTI4_IRQn	EXTI Line 4 interrupt
DMA0_Channel0_IRQn	DMA0 Channel0 global interrupt
DMA0_Channel1_IRQn	DMA0 Channel1 global interrupt
DMA0_Channel2_IRQn	DMA0 Channel2 global interrupt
DMA0_Channel3_IRQn	DMA0 Channel3 global interrupt
DMA0_Channel4_IRQn	DMA0 Channel4 global interrupt
DMA0_Channel5_IRQn	DMA0 Channel5 global interrupt

Member name	Function description
DMA0_Channel6_IRQn	DMA0 Channel6 global interrupt
ADC0_1_IRQn	ADC0 and ADC1 interrupt
EXTI5_9_IRQn	EXTI Line5-9 interrupts
TIMER0_BRK_IRQn	TIMER0 break interrupt
TIMER0_UP_IRQn	TIMER0 update interrupt
TIMER0_TRG_CMT_IDX_IRQn	TIMER0 trigger and commutation interrupt/TIMER0 direction change interrupt/TIMER0 index
TIMER0_Channel_IRQn	TIMER0 channel capture compare interrupt
TIMER1_IRQn	TIMER1 global interrupt
TIMER2_IRQn	TIMER2 global interrupt
TIMER3_IRQn	TIMER3 global interrupt
I2C0_EV_WKUP_IRQn	I2C0 event and wakeup interrupt
I2C0_ER_IRQn	I2C0 error interrupt
I2C1_EV_WKUP_IRQn	I2C1 event and wakeup interrupt
I2C1_ER_IRQn	I2C1 error interrupt
SPI0_IRQn	SPI0 global interrupt
SPI1_IRQn	SPI1 global interrupt
USART0_IRQn	USART0 global interrupt and wakeup through EXTI line28 interrupt
USART1_IRQn	USART1 global interrupt and wakeup through EXTI line29 interrupt
USART2_IRQn	USART2 global interrupt and wakeup through EXTI line30 interrupt
EXTI10_15_IRQn	EXTI Line10-15 interrupts
RTC_Alarm_IRQn	RTC alarm from EXTI line17 interrupt
TIMER7_BRK_TRS_IDX_IRQHandler	TIMER7 break interrupt/ transition error/ index error
TIMER7_UP_IRQn	TIMER7 update interrupt
TIMER7_TRG_CMT_IDX_IRQHandler	TIMER7 trigger and commutation interrupt/direction change interrupt/index
TIMER7_Channel_IRQn	TIMER7 capture compare interrupt
ADC2_IRQHandler	ADC2 global interrupt
SYSCFG_IRQHandler	SYSCFG interrupt
LPTIMER_IRQHandler	LPTIMER global interrupt
TIMER4_IRQHandler	TIMER4 global interrupt
SPI2_IRQHandler	SPI2 global interrupt
UART3_IRQHandler	UART3 global interrupt
UART4_IRQHandler	UART4 global interrupt
TIMER5_DAC0_2_UDR_IRQHandler	TIMER5 global interrupt and DAC2 / DAC0 underrun error interrupt
TIMER6_DAC1_3_UDR_IRQHandler	TIMER6 global interrupt and DAC3 / DAC1 underrun error interrupt

Member name	Function description
DMA1_Channel0_IRQHandler	DMA1 channel0 global interrupt
DMA1_Channel1_IRQHandler	DMA1 channel1 global interrupt
DMA1_Channel2_IRQHandler	DMA1 channel2 global interrupt
DMA1_Channel3_IRQHandler	DMA1 channel3 global interrupt
DMA1_Channel4_IRQHandler	DMA1 channel4 global interrupt
ADC3_IRQHandler	ADC3 global interrupt
VUVD1_VOVD1_IRQHandler	VUVD1 / VOVD1 interrupt
CMP0_3_IRQHandler	CMP0 / CMP1 / CMP2 / CMP3 through EXTI lines 20 / 21 / 22 / 23 interrupts
CMP4_7_IRQHandler	CMP4 / CMP5 / CMP6 / CMP7 through EXTI lines 24 / 36 / 37 / 38 interrupts
CMP_IRQHandler	CMP global interrupt
HRTIMER_IRQ0_IRQHandler	HRTIMER interrupt0
HRTIMER_IRQ1_IRQHandler	HRTIMER interrupt1
HRTIMER_IRQ2_IRQHandler	HRTIMER interrupt2
HRTIMER_IRQ3_IRQHandler	HRTIMER interrupt3
HRTIMER_IRQ4_IRQHandler	HRTIMER interrupt4
HRTIMER_IRQ5_IRQHandler	HRTIMER interrupt5
HRTIMER_IRQ6_IRQHandler	HRTIMER interrupt6
HRTIMER_IRQ7_IRQHandler	HRTIMER interrupt7
HRTIMER_IRQ8_IRQHandler	HRTIMER interrupt8
HRTIMER_IRQ9_IRQHandler	HRTIMER interrupt9
TIMER19_BRK_TRS_IDX_IRQHandler	TIMER19 break interrupt / TIMER19 transition error / TIMER19 Index error
TIMER19_UP_IRQHandler	TIMER19 update interrupt
TIMER19_TRG_CMT_IDX_IRQHandler	TIMER19 trigger and commutation interrupt / TIMER19 direction change interrupt / TIMER19 index
TIMER19_Channel_IRQHandler	TIMER19 capture compare interrupt
FPU_IRQHandler	FPU global interrupt
I2C2_EV_WKUP_IRQHandler	I2C2 event interrupt and I2C2 wakeup through EXTI interrupt
I2C2_ER_IRQHandler	I2C2 error interrupt
CAU_IRQHandler	CAU global interrupt
TRNG_IRQHandler	TRNG global interrupt
I2C3_EV_WKUP_IRQHandler	I2C3 event interrupt and I2C3 wakeup through EXTI line34 interrupt
I2C3_ER_IRQHandler	I2C3 error interrupt
DMAMUX_OVR_IRQHandler	DMAMUX overrun interrupt
QSPI_IRQHandler	QSPI global interrupt
FFT_IRQHandler	FFT global interrupt
DMA1_Channel5_IRQHandler	DMA1 channel5 global interrupt

Member name	Function description
DMA1_Channel6_IRQHandler	DMA1 channel6 global interrupt
CLA_IRQHandler	CLA interrupt
TMU_IRQHandler	TMU interrupt
FAC_IRQHandler	FAC interrupt
HPDF_INT0_IRQHandler	HPDF global interrupt 0
HPDF_INT1_IRQHandler	HPDF global interrupt 1
HPDF_INT2_IRQHandler	HPDF global interrupt 2
HPDF_INT3_IRQHandler	HPDF global interrupt 3
TIMER14_IRQHandler	TIMER14 global interrupt
TIMER15_IRQHandler	TIMER15 global interrupt
TIMER16_IRQHandler	TIMER16 global interrupt
CAN0_WKUP_IRQHandler	CAN0 wakeup through EXTI line25 interrupt
CAN0_Message_IRQHandler	CAN0 interrupt for message buffer
CAN0_Busoff_IRQHandler	CAN0 interrupt for bus off / bus off done
CAN0_Error_IRQHandler	CAN0 interrupt for error
CAN0_FastError_IRQHandler	CAN0 interrupt for error in fast transmission
CAN0_TEC_IRQHandler	CAN0 interrupt for transmit warning
CAN0_REC_IRQHandler	CAN0 interrupt for receive warning
CAN1_WKUP_IRQHandler	CAN1 wakeup through EXTI line26 interrupt
CAN1_Message_IRQHandler	CAN1 interrupt for message buffer
CAN1_Busoff_IRQHandler	CAN1 interrupt for bus off / bus off done
CAN1_Error_IRQHandler	CAN1 interrupt for error
CAN1_FastError_IRQHandler	CAN1 interrupt for error in fast transmission
CAN1_TEC_IRQHandler	CAN1 interrupt for transmit warning
CAN1_REC_IRQHandler	CAN1 interrupt for receive warning
CAN2_WKUP_IRQHandler	CAN2 wakeup through EXTI line27 interrupt
CAN2_Message_IRQHandler	CAN2 interrupt for message buffer
CAN2_Busoff_IRQHandler	CAN2 interrupt for bus off / bus off done
CAN2_Error_IRQHandler	CAN2 interrupt for error
CAN2_FastError_IRQHandler	CAN2 interrupt for error in fast transmission
CAN2_TEC_IRQHandler	CAN2 interrupt for transmit warning
CAN2_REC_IRQHandler	CAN2 interrupt for receive warning
TIMER0_DEC_IRQHandler	TIMER0 DEC interrupt
TIMER1_DEC_IRQHandler	TIMER1 DEC interrupt
TIMER2_DEC_IRQHandler	TIMER2 DEC interrupt
TIMER3_DEC_IRQHandler	TIMER3 DEC interrupt
TIMER4_DEC_IRQHandler	TIMER4 DEC interrupt
TIMER7_DEC_IRQHandler	TIMER7 DEC interrupt
TIMER19_DEC_IRQHandler	TIMER19 DEC interrupt

## nvic\_priority\_group\_set

The description of nvic\_priority\_group\_set is shown as below:

**Table 3-748. Function nvic\_priority\_group\_set**

<b>Function name</b>	nvic_priority_group_set
<b>Function prototype</b>	void nvic_priority_group_set(uint32_t nvic_prigroup);
<b>Function descriptions</b>	set the priority group
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>nvic_prigroup</b>	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 E3_SUB1	3 bits for pre-emption priority 1 bits for subpriority
NVIC_PRIGROUP_PR E4_SUB0	4 bits for pre-emption priority 0 bits for subpriority
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-749. Function nvic\_irq\_enable**

<b>Function name</b>	nvic_irq_enable
<b>Function prototype</b>	void nvic_irq_enable(uint8_t nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t nvic_irq_sub_priority);
<b>Function descriptions</b>	enable NVIC interrupt request
<b>Precondition</b>	-
<b>The called functions</b>	nvic_priority_group_set
<b>Input parameter{in}</b>	
<b>nvic_irq</b>	NVIC interrupt, refer to <a href="#">Table 3-747. Enum IRQn_Type</a>

Input parameter{in}	
<b>nvic_irq_pre_priority</b>	the pre-emption priority needed to set
Input parameter{in}	
<b>nvic_irq_sub_priority</b>	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 3-750. Function nvic\_irq\_disable**

<b>Function name</b>	nvic_irq_disable
<b>Function prototype</b>	void nvic_irq_disable(uint8_t nvic_irq);
<b>Function descriptions</b>	disable NVIC interrupt request
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>nvic_irq</b>	NVIC interrupt, refer to enum <a href="#">Table 3-747. Enum IRQn_Type</a>
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 3-751. Function nvic\_system\_reset**

<b>Function name</b>	nvic_system_reset
<b>Function prototype</b>	void nvic_system_reset(void);
<b>Function descriptions</b>	initiates a system reset request to reset the MCU
<b>Precondition</b>	-
<b>The called functions</b>	-



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

Example:

```
/* initiates a system reset request to reset the MCU */
nvic_system_reset();
```

### nvic\_vector\_table\_set

The description of nvic\_vector\_table\_set is shown as below:

**Table 3-752. 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
NVIC_VECTTAB_RAM	RAM base address
NVIC_VECTTAB_FLASH	Flash base address
Input parameter{in}	
offset	vector table 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 3-753. Function system\_lowpower\_set**

Function name	system_lowpower_set
Function prototype	void system_lowpower_set(uint8_t lowpower_mode);

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

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 3-754. Function system\_lowpower\_reset**

<b>Function name</b>	system_lowpower_reset
<b>Function prototype</b>	void system_lowpower_reset(uint8_t lowpower_mode);
<b>Function descriptions</b>	reset the state of the low power mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lowpower_mode</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* the system will exit low power mode by exiting from ISR */
system_lowpower_reset(SCB_LPM_SLEEP_EXIT_ISR);
```

### systick\_cksourceset

The description of systick\_cksourceset is shown as below:

**Table 3-755. Function systick\_cksourceset**

<b>Function name</b>	systick_cksourceset
<b>Function prototype</b>	void systick_cksourceset(uint32_t systick_cksourceset);
<b>Function descriptions</b>	set the systick clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>systick_cksourceset</b>	the systick clock source needed to choose
<i>SYSTICK_CLKSOURCE_0</i>	systick clock source is from CK_SYS
<i>SYSTICK_CLKSOURCE_1</i>	systick clock source is from CK_SYS / 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* systick clock source is CK_SYS / 2 */
systick_cksourceset(SYSTICK_CLKSOURCE_1);
```

### mpu\_enable

The description of mpu\_enable is shown as below:

**Table 3-756. Function mpu\_enable**

<b>Function name</b>	mpu_enable
<b>Function prototype</b>	void mpu_enable(uint32_t MPU_Control);
<b>Function descriptions</b>	enable the MPU
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>MPU_Control</b>	select a different MPU mode
<i>MPU_MODE_HFNMI_PRIVDEF_NONE</i>	HFNMIENA and PRIVDEFENA are 0
<i>MPU_MODE_HFNMI_PRIVDEF_HARDFAULT</i>	use the MPU for memory accesses by HardFault and NMI handlers only

<i>MPU_MODE_PRIV_DEFAULT</i>	enables the default memory map as a background region for privileged access only
<i>MPU_MODE_HFNMI_PRIVDEF</i>	HFNMENA and PRIVDEFENA are 1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the MPU */
mpu_enable(MPU_MODE_PRIV_DEFAULT);
```

### mpu\_disable

The description of mpu\_disable is shown as below:

**Table 3-757. Function mpu\_disable**

<b>Function name</b>	mpu_disable
<b>Function prototype</b>	void mpu_disable(void);
<b>Function descriptions</b>	disable the MPU
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the MPU */
mpu_disable();
```

### mpu\_region\_struct\_para\_init

The description of mpu\_region\_struct\_para\_init is shown as below:

**Table 3-758. Function mpu\_region\_struct\_para\_init**

<b>Function name</b>	mpu_region_struct_para_init
<b>Function prototype</b>	void mpu_region_struct_para_init(mpu_region_init_struct *mpu_init_struct);
<b>Function descriptions</b>	initialize mpu_region_init_struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
mpu_init_struct	MPU initialization structure, refer to <a href="#">Table 3-745. Structure mpu_region_init_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
mpu_region_init_struct mpu_init_para;

/* initialize mpu_region_init_struct with the default values */

mpu_region_struct_para_init(&mpu_init_para);
```

### mpu\_attribute\_struct\_para\_init

The description of mpu\_attribute\_struct\_para\_init is shown as below:

**Table 3-759. Function mpu\_attribute\_struct\_para\_init**

Function name	mpu_attribute_struct_para_init
Function prototype	void mpu_attribute_struct_para_init(mpu_attribute_init_struct *attribute_init_struct);
Function descriptions	initialize mpu_attribute_init_struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
mpu_attribute_init_struct	MPU attribute initialization structure, refer to <a href="#">Table 3-746. Structure mpu_attribute_init_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
mpu_attribute_init_struct mpu_attribute_init_para;

/* initialize mpu_attribute_init_struct with the default values */

mpu_attribute_struct_para_init(&mpu_attribute_init_para);
```

### mpu\_region\_config

The description of mpu\_region\_config is shown as below:

**Table 3-760. Function mpu\_region\_config**

Function name	mpu_region_config
---------------	-------------------

<b>Function prototype</b>	void mpu_region_config(mpu_region_init_struct *mpu_init_struct);
<b>Function descriptions</b>	configure the MPU region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mpu_init_struct</b>	MPU initialization structure, refer to <a href="#">Table 3-745. Structure mpu_region_init_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

mpu_region_init_struct region_init_struct;

mpu_region_struct_para_init(&region_init_struct);

region_init_struct.region_number = MPU_REGION_NUMBER0;

region_init_struct.region_base_address = 0x20002000U;

region_init_struct.region_limit_address = 0x20002000U;

region_init_struct.access_permission = MPU_REGION_PRIVILEGED_RO;

region_init_struct.shareability = MPU_ACCESS_NOT_SHAREABLE;

region_init_struct.instruction_exec = MPU_INSTRUCTION_EXEC_PERMIT;

region_init_struct.attribute_index = MPU_ATTRIBUTE_NUMBER4;

/* configure the MPU region */

mpu_region_config(&region_init_struct);

```

### mpu\_attribute\_config

The description of mpu\_attribute\_config is shown as below:

**Table 3-761. Function mpu\_attribute\_config**

<b>Function name</b>	mpu_attribute_config
<b>Function prototype</b>	void mpu_attribute_config(mpu_attribute_init_struct *attribute_init_struct);
<b>Function descriptions</b>	configure the MPU attribute
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mpu_attribute_init_struct</b>	MPU attribute initialization structure, refer to <a href="#">Table 3-745. Structure mpu_region_init_struct</a> .
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```

mpu_attribute_init_struct attribute_init_struct;

mpu_attribute_struct_para_init(&attribute_init_struct);

attribute_init_struct.attribute_number = MPU_ATTRIBUTE_NUMBER4;

attribute_init_struct.memory_type = MPU_MEMORY_NORMAL;

attribute_init_struct.outer_attributes = MPU_NORMAL_OUTER_WT_TRAN_RW_ALLOC;

attribute_init_struct.inner_attributes = MPU_NORMAL_INNER_WT_TRAN_RW_ALLOC;

/* configure the MPU attribute */

mpu_attribute_config(&attribute_init_struct);

```

### mpu\_region\_enable

The description of mpu\_region\_enable is shown as below:

**Table 3-762. Function mpu\_region\_enable**

<b>Function name</b>	mpu_region_enable
<b>Function prototype</b>	void mpu_region_enable(void);
<b>Function descriptions</b>	enable MPU region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* enable MPU region */

mpu_region_enable();

```

## 3.23. 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 [3.23.1](#),

the PMU firmware functions are introduced in chapter [3.23.2](#).

### 3.23.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

**Table 3-763. PMU Registers**

Registers	Descriptions
PMU_CTL0	PMU control register 0
PMU_CS	PMU control and status register
PMU_CTL1	PMU control register 1
PMU_CTL2	PMU control register 2
PMU_CTL3	PMU control register 3

### 3.23.2. Descriptions of Peripheral functions

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

**Table 3-764. PMU firmware function**

Function name	Function description
pmu_deinit	reset PMU registers
pmu_lvd_select	select low voltage detector threshold
pmu_lvd_enable	enable PMU lvd
pmu_lvd_disable	disable PMU lvd
pmu_avd_select	select analog voltage detector threshold
pmu_avd_enable	enable PMU analog voltage detector
pmu_avd_disable	disable PMU analog voltage detector
pmu_ovd_select	select PMU core over voltage detector threshold
pmu_ovd_enable	enable PMU core over voltage detector
pmu_ovd_disable	disable PMU core over voltage detector
pmu_uvd_select	select PMU core under voltage detector threshold
pmu_uvd_enable	enable PMU core under voltage detector
pmu_uvd_disable	disable PMU core under voltage detector
pmu_ovd_filter	PMU core over voltage digital noise filter
pmu_uvd_filter	PMU core under voltage digital noise filter
pmu_deepsleep_voltage	PMU deepsleep mode voltage selection
pmu_vbat_charging_select	PMU VBAT battery charging resistor selection
pmu_vbat_charging_enable	enable VBAT battery charging
pmu_vbat_charging_disable	disable VBAT battery charging
pmu_vbat_temp_monitor_enable	VBAT and temperature monitoring enable
pmu_vbat_temp_monitor_disable	disable VBAT and temperature monitoring
pmu_to_sleepmode	enter sleep mode
pmu_to_deepsleepmode	enter deepsleep mode
pmu_to_standbymode	enter 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 backup domain write
pmu_backup_write_disable	disable backup domain write
pmu_flag_get	get flag state
pmu_flag_clear	clear flag bit

## pmu\_deinit

The description of pmu\_deinit is shown as below:

**Table 3-765. 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 register */
```

```
pmu_deinit();
```

## pmu\_lvd\_select

The description of pmu\_lvd\_select is shown as below:

**Table 3-766. 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 2.15V
PMU_LVDT_1	voltage threshold is 2.3V
PMU_LVDT_2	voltage threshold is 2.45V

<i>PMU_LVDT_3</i>	voltage threshold is 2.6V
<i>PMU_LVDT_4</i>	voltage threshold is 2.75V
<i>PMU_LVDT_5</i>	voltage threshold is 2.9V
<i>PMU_LVDT_6</i>	voltage threshold is 3.0V
<i>PMU_LVDT_7</i>	input analog voltage on PA10 (compared with 1.2V)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select low voltage detector threshold as 3.0V */
```

```
pmu_lvd_select(PMU_LVDT_6);
```

### pmu\_lvd\_enable

The description of pmu\_lvd\_enable is shown as below:

**Table 3-767. Function pmu\_lvd\_enable**

<b>Function name</b>	pmu_lvd_enable
<b>Function prototype</b>	void pmu_lvd_enable(void);
<b>Function descriptions</b>	enable PMU lvd
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable PMU lvd */
```

```
pmu_lvd_enable();
```

### pmu\_lvd\_disable

The description of pmu\_lvd\_disable is shown as below:

**Table 3-768. Function pmu\_lvd\_disable**

<b>Function name</b>	pmu_lvd_disable
<b>Function prototype</b>	void pmu_lvd_disable(void);
<b>Function descriptions</b>	disable PMU lvd

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

Example:

```
/* disable PMU lvd */
```

```
pmu_lvd_disable();
```

### pmu\_avd\_select

The description of pmu\_avd\_select is shown as below:

**Table 3-769. Function pmu\_avd\_select**

Function name	pmu_avd_select
Function prototype	void pmu_avd_select(uint32_t avdt_n);
Function descriptions	select analog voltage detector threshold
Precondition	-
The called functions	-
Input parameter{in}	
avdt_n	select analog voltage detector threshold
PMU_AVDT_0	voltage threshold of analog voltage detector is 1.8V
PMU_AVDT_1	voltage threshold of analog voltage detector is 2.2V
PMU_AVDT_2	voltage threshold of analog voltage detector is 2.6V
PMU_AVDT_3	voltage threshold of analog voltage detector is 2.9V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select analog voltage detector threshold 2.9V */
```

```
pmu_avd_select(PMU_VAVDVC_3);
```

### pmu\_avd\_enable

The description of pmu\_avd\_enable is shown as below:

Table 3-770. Function pmu\_avd\_enable

Function name	pmu_avd_enable
Function prototype	void pmu_avd_enable(void);
Function descriptions	enable PMU analog voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable PMU analog voltage detector */
```

```
pmu_avd_enable();
```

### pmu\_avd\_disable

The description of pmu\_avd\_disable is shown as below:

Table 3-771. Function pmu\_avd\_disable

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

Example:

```
/* disable PMU analog voltage detector */
```

```
pmu_avd_disable();
```

### pmu\_ovd\_select

The description of pmu\_ovd\_select is shown as below:

Table 3-772. Function pmu\_ovd\_enable

Function name	pmu_ovd_select
Function prototype	void pmu_ovd_select (uint32_t ovdn);
Function descriptions	enable PMU V <sub>CORE</sub> voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
ovdn	voltage threshold value
PMU_OVDN_0	oltage threshold is 1.25V
PMU_OVDN_1	voltage threshold is 1.35V
PMU_OVDN_2	voltage threshold is 1.45V
PMU_OVDN_3	voltage threshold is 1.55V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select PMU core over voltage detector threshold 1.25V */
```

```
pmu_ovd_select (PMU_OVDN_0);
```

### pmu\_ovd\_enable

The description of pmu\_ovd\_enable is shown as below:

Table 3-773. Function pmu\_ovd\_enable

Function name	pmu_cvd_enable
Function prototype	void pmu_ovd_enable(void);
Function descriptions	enable PMU V <sub>CORE</sub> voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable PMU VCORE voltage detector */
```

```
pmu_ovd_enable();
```

## pmu\_ovd\_disable

The description of pmu\_ovd\_disable is shown as below:

**Table 3-774. Function pmu\_ovd\_disable**

<b>Function name</b>	pmu_cvd_disable
<b>Function prototype</b>	void pmu_cvd_disable(void);
<b>Function descriptions</b>	disable PMU V <sub>0.9V</sub> core voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable PMU V0.9V voltage detector */
pmu_cvd_disable();
```

## pmu\_uvd\_select

The description of pmu\_uvd\_select is shown as below:

**Table 3-775. Function pmu\_uvd\_select**

<b>Function name</b>	pmu_uvd_select
<b>Function prototype</b>	void pmu_uvd_select (uint32_t ldo_n);
<b>Function descriptions</b>	select PMU core under voltage detector threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ldo_n</b>	Select LDO output voltage
<i>PMU_UVDT_0</i>	LDO output voltage 0.65V mode
<i>PMU_UVDT_1</i>	LDO output voltage 0.75V mode
<i>PMU_UVDT_2</i>	LDO output voltage 0.85V mode
<i>PMU_UVDT_3</i>	LDO output voltage 0.95V mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select LDO output voltage 0.95V */
```

```
pmu_uvd_select(PMU_LDOVS_3);
```

## pmu\_uvd\_enable

The description of pmu\_uvd\_enable is shown as below:

**Table 3-776. Function pmu\_uvd\_enable**

<b>Function name</b>	pmu_uvd_enable
<b>Function prototype</b>	void pmu_uvd_enable (void);
<b>Function descriptions</b>	enable PMU core under voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable PMU core under voltage detector */
pmu_uvd_enable();
```

## pmu\_uvd\_disable

The description of pmu\_uvd\_disable is shown as below:

**Table 3-777. Function pmu\_uvd\_disable**

<b>Function name</b>	pmu_uvd_disable
<b>Function prototype</b>	void pmu_uvd_disable (void);
<b>Function descriptions</b>	disable PMU core under voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable PMU core under voltage detector */
pmu_uvd_disable();
```

## pmu\_ovd\_filter

The description of pmu\_ovd\_filter is shown as below:

**Table 3-778. Function pmu\_ovd\_filter**

<b>Function name</b>	pmu_ovd_filter
<b>Function prototype</b>	void pmu_ovd_filter (uint32_t dnf);
<b>Function descriptions</b>	PMU core over voltage digital noise filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dnf</b>	digital noise filter
<i>Uint32_t</i>	0~255
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* Turn on digital filter, filter peak length up to 255 * 1024 * TPCLK r */
```

```
pmu_ovd_filter(255);
```

## pmu\_uvd\_filter

The description of pmu\_uvd\_filter is shown as below:

**Table 3-779. Function pmu\_uvd\_filter**

<b>Function name</b>	pmu_uvd_filter
<b>Function prototype</b>	void pmu_uvd_filter (uint32_t dnf);
<b>Function descriptions</b>	PMU core under voltage digital noise filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dnf</b>	digital noise filter
<i>uint32_t</i>	0~255
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* close under voltage digital noise filter */
```

```
pmu_uvd_filter(0);
```



## pmu\_deepsleep\_voltage

The description of pmu\_deepsleep\_voltage is shown as below:

**Table 3-780. Function pmu\_deepsleep\_voltage**

<b>Function name</b>	pmu_deepsleep_voltage
<b>Function prototype</b>	void pmu_deepsleep_voltage (uint32_t vol);
<b>Function descriptions</b>	PMU deepsleep mode voltage selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvdt_n</b>	voltage threshold value
<i>PMU_DSLPVS_0</i>	deepsleep mode voltage 0.8V
<i>PMU_DSLPVS_1</i>	deepsleep mode voltage 0.9V
<i>PMU_DSLPVS_2</i>	deepsleep mode voltage 1.0V (default)
<i>PMU_DSLPVS_3</i>	deepsleep mode voltage 1.1V
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select deepsleep mode voltage 0.8V */
pmu_deepsleep_voltage (PMU_DSLPVS_0);
```

## pmu\_vbat\_charging\_select

The description of pmu\_vbat\_charging\_select is shown as below:

**Table 3-781. Function pmu\_vbat\_charging\_select**

<b>Function name</b>	pmu_vbat_charging_select
<b>Function prototype</b>	void pmu_vbat_charging_select(uint32_t resistor);
<b>Function descriptions</b>	PMU V <sub>BAT</sub> battery charging resistor selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>resistor</b>	select PMU VBAT battery charging resistor
<i>PMU_VCRSEL_5K</i>	5 kOhms resistor is selected for charging VBAT battery
<i>PMU_VCRSEL_1P5K</i>	1.5 kOhms resistor is selected for charging VBAT battery
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select PMU VBAT battery charging resistor to 1.5 kOhms */
pmu_vbat_charging_select(PMU_VCRSEL_1P5K);
```

### pmu\_vbat\_charging\_enable

The description of pmu\_vbat\_charging\_enable is shown as below:

**Table 3-782. Function pmu\_vbat\_charging\_enable**

<b>Function name</b>	pmu_vbat_charging_enable
<b>Function prototype</b>	void pmu_vbat_charging_enable(void);
<b>Function descriptions</b>	enable V <sub>BAT</sub> battery charging
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable VBAT battery charging */
pmu_vbat_charging_enable();
```

### pmu\_vbat\_charging\_disable

The description of pmu\_vbat\_charging\_disable is shown as below:

**Table 3-783. Function pmu\_vbat\_charging\_disable**

<b>Function name</b>	pmu_vbat_charging_disable
<b>Function prototype</b>	void pmu_vbat_charging_disable(void);
<b>Function descriptions</b>	disable V <sub>BAT</sub> battery charging
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable VBAT battery charging */
```

```
pmu_vbat_charging_disable();
```

### pmu\_vbat\_temp\_monitor\_enable

The description of pmu\_vbat\_temp\_monitor\_enable is shown as below:

**Table 3-784. Function pmu\_vbat\_temp\_monitor\_enable**

<b>Function name</b>	pmu_vbat_temp_monitor_enable
<b>Function prototype</b>	void pmu_vbat_temp_monitor_enable(void);
<b>Function descriptions</b>	enable V <sub>BAT</sub> and temperature monitoring
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable VBAT and temperature monitoring */
```

```
pmu_vbat_temp_monitor_enable();
```

### pmu\_vbat\_temp\_monitor\_disable

The description of pmu\_vbat\_temp\_monitor\_disable is shown as below:

**Table 3-785. Function pmu\_vbat\_temp\_monitor\_disable**

<b>Function name</b>	pmu_vbat_temp_monitor_disable
<b>Function prototype</b>	void pmu_vbat_temp_monitor_disable(void);
<b>Function descriptions</b>	disable V <sub>BAT</sub> and temperature monitoring
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable VBAT and temperature monitoring */
```

```
pmu_vbat_temp_monitor_disable();
```

## pmu\_to\_sleepmode

The description of pmu\_to\_sleepmode is shown as below:

**Table 3-786. Function pmu\_to\_sleepmode**

<b>Function name</b>	pmu_to_sleepmode
<b>Function prototype</b>	void pmu_to_sleepmode(uint8_t sleepmodecmd);
<b>Function descriptions</b>	enter sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sleepmodecmd</b>	command to enter sleep mode
WFI_CMD	use WFI command
WFE_CMD	use WFE command
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-787. Function pmu\_to\_deepsleepmode**

<b>Function name</b>	pmu_to_deepsleepmode
<b>Function prototype</b>	void pmu_to_deepsleepmode(uint32_t lowpower, uint8_t deepsleepmodecmd)
<b>Function descriptions</b>	enter deepsleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lowpower</b>	LDO lowpower mode
PMU_LDO_NORMAL	LDO normal work when pmu enter deepsleep mode
PMU_LDO_LOWPOWER	LDO work at low power mode when pmu enter deepsleep mode
<b>Input parameter{in}</b>	
<b>deepsleepmodecmd</b>	command to enter deepsleep mode

WFI_CMD	use WFI command
WFE_CMD	use WFE command
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PMU work in deepsleep mode */
```

```
pmu_to_deepsleepmode(WFI_CMD);
```

### pmu\_to\_standbymode

The description of pmu\_to\_standbymode is shown as below:

**Table 3-788. Function pmu\_to\_standbymode**

Function name	pmu_to_standbymode
Function prototype	void pmu_to_standbymode(void);
Function descriptions	enter 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 3-789. 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 PMU wakeup pin
Precondition	-
The called functions	-
Input parameter{in}	

<b>wakeup_pin</b>	Wakeup pin
<i>PMU_WAKEUP_PIN0</i>	WKUP Pin 0 (PA0)
<i>PMU_WAKEUP_PIN1</i>	WKUP Pin 1 (PC13)
<i>PMU_WAKEUP_PIN2</i>	WKUP Pin 2 (PE6)
<i>PMU_WAKEUP_PIN3</i>	WKUP Pin 3 (PA2)
<i>PMU_WAKEUP_PIN4</i>	WKUP Pin 4 (PC5)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-790. Function pmu\_wakeup\_pin\_disable**

<b>Function name</b>	pmu_wakeup_pin_disable
<b>Function prototype</b>	void pmu_wakeup_pin_disable(uint32_t wakeup_pin);
<b>Function descriptions</b>	disable PMU wakeup pin
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wakeup_pin</b>	Wakeup pin
<i>PMU_WAKEUP_PIN0</i>	WKUP Pin 0 (PA0)
<i>PMU_WAKEUP_PIN1</i>	WKUP Pin 1 (PA2)
<i>PMU_WAKEUP_PIN3</i>	WKUP Pin 3 (PC13)
<i>PMU_WAKEUP_PIN5</i>	WKUP Pin 5 (PC1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-791. Function pmu\_backup\_write\_enable**

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

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 3-792. Function pmu\_backup\_write\_disable**

<b>Function name</b>	pmu_backup_write_disable
<b>Function prototype</b>	void pmu_backup_write_disable(void);
<b>Function descriptions</b>	disable backup domain write
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### pmu\_flag\_get

The description of pmu\_flag\_get is shown as below:

Table 3-793. Function `pmu_flag_get`

<b>Function name</b>	<code>pmu_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus pmu_flag_get(uint32_t flag);</code>
<b>Function descriptions</b>	get flag state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	flag
<code>PMU_FLAG_WAKEUP</code>	wakeup flag
<code>PMU_FLAG_STANDBY</code>	standby flag
<code>PMU_FLAG_LVDF</code>	low voltage detector status flag
<code>PMU_FLAG_VAVDF</code>	$V_{DDA}$ analog voltage detector voltage output on $V_{DDA}$ flag
<code>PMU_FLAG_VOVDFO</code>	Over voltage on $V_{DDA}$ detector flag
<code>PMU_FLAG_VUVDFO</code>	Under voltage on $V_{DDA}$ detector flag
<code>PMU_FLAG_VOVDFO1</code>	Over voltage on $V_{DDA}$ detector flag after digital filter
<code>PMU_FLAG_VUVDFO1</code>	Under voltage on $V_{DDA}$ detector flag after digital filter
<code>PMU_FLAG_BKPVSFRF</code>	peripheral voltage on $V_{DDA}$ detector flag
<code>PMU_FLAG_VBATLF</code>	VBAT level monitoring versus low threshold
<code>PMU_FLAG_VBATHF</code>	VBAT level monitoring versus high threshold
<code>PMU_FLAG_TEMPLF</code>	temperature level monitoring versus low threshold
<code>PMU_FLAG_TEMPHE</code>	temperature level monitoring versus high threshold
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-794. Function `pmu_flag_clear`

<b>Function name</b>	<code>pmu_flag_clear</code>
<b>Function prototype</b>	<code>void pmu_flag_clear(uint32_t flag);</code>
<b>Function descriptions</b>	clear flag bit
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	



flag	flag
<i>PMU_FLAG_WAKEUP</i>	wakeup flag
<i>PMU_FLAG_STANDBY</i>	standby flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear flag bit */
```

```
pmu_flag_clear(PMU_FLAG_STANDBY);
```

## 3.24. QSPI

The QSPI is a specialized interface that communicate with Flash memories. This interface support single, dual or quad SPI FLASH. The QSPI registers are listed in chapter [3.24.1](#), the QSPI firmware functions are introduced in chapter [3.24.2](#).

### 3.24.1. Descriptions of Peripheral registers

QSPI registers are listed in the table shown as below:

**Table 3-795. QSPI registers**

Registers	Descriptions
QSPI_CTL	QSPI control register
QSPI_DCFG	QSPI device configuration register
QSPI_STAT	QSPI status register
QSPI_STATC	QSPI status clear register
QSPI_DTLEN	QSPI data length register
QSPI_TCFG	QSPI transfer configuration register
QSPI_ADDR	QSPI address register
QSPI_ALTE	QSPI alternate bytes register
QSPI_DATA	QSPI data register
QSPI_STATMK	QSPI status mask register
QSPI_STATMATCH	QSPI status match register
QSPI_INTERVAL	QSPI interval register
QSPI_TMOUT	QSPI timeout register
QSPI_FLUSH	QSPI fifo flush register

### 3.24.2. Descriptions of Peripheral functions

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

Table 3-796. QSPI firmware function

Function name	Function description
qspi_deinit	reset QSPI peripheral
qspi_struct_para_init	initialize the parameters of QSPI init structure with the default values
qspi_cmd_struct_para_init	initialize the parameters of QSPI command structure with the default values
qspi_polling_struct_para_init	initialize the parameters of QSPI read polling structure with the default values
qspi_init	initialize QSPI
qspi_enable	enable QSPI
qspi_disable	disable QSPI
qspi_dma_enable	enable QSPI DMA
qspi_dma_disable	disable QSPI DMA
qspi_command_config	configure QSPI command parameters
qspi_polling_config	configure QSPI read polling mode
qspi_memorymapped_config	configure QSPI memory mapped mode
qspi_data_transmit	QSPI transmit data function
qspi_data_receive	QSPI receive data function
qspi_transmission_abort	abort the current transmission
qspi_output_clock_delay_enable	enable output clock delay
qspi_output_clock_delay_disable	disable output clock delay
qspi_output_clock_delay_config	configure output clock delay
qspi_sample_shift_config	configure QSPI sample shift
qspi_receive_clock_sel	select QSPI receive clock
qspi_delay_scan_enable	enable delay scan
qspi_delay_scan_disable	disable delay scan
qspi_csn_edge_cycle	select csn falls and rises 1 or 2 sck cycles
qspi_flag_get	get QSPI flag status
qspi_flag_clear	clear QSPI flag status
qspi_interrupt_enable	enable QSPI interrupt
qspi_interrupt_disable	disable QSPI interrupt
qspi_interrupt_flag_get	get QSPI interrupt flag status
qspi_interrupt_flag_clear	clear QSPI interrupt flag status

## Structure qspi\_init\_struct

Table 3-797. Structure qspi\_init\_struct

Member name	Function description
prescaler	QSPI prescaler
fifo_threshold	QSPI FIFO threshold
sample_shift1	QSPI sample shift 1
sample_shift2	QSPI sample shift 2

Member name	Function description
flash_size	external flash size
cs_high_time	CLK cycles which the chip select (nCS) must stay high between two command sequences
clock_mode	QSPI clock mode

### Structure qspi\_command\_struct

**Table 3-798. Structure qspi\_command\_struct**

Member name	Function description
instruction_mode	instruction mode
instruction	8 bits instruction
addr_mode	address mode
addr_size	address size
addr	address to be send to the external flash memory
altebytes_mode	alternate bytes mode
altebytes_size	alternate bytes size
altebytes	alternate bytes information
dummycycles	dummy cycles
data_mode	data mode
data_length	32 bits data length
sioo_mode	send instruction only once mode
trans_rate	specifies the transfer rate is SDR or DDR mode
trans_delay	specifies the transfer delay is 0 or 1/4 cycle

### Structure qspi\_polling\_struct

**Table 3-799. Structure qspi\_polling\_struct**

Member name	Function description
match	match value
mask	mask value
interval	number of clock cycles between two read during automatic polling phases
statusbytes_size	the size of the status bytes received
match_mode	method used for determining a match
polling_stop	if read polling is stopped after a match

### qspi\_deinit

The description of qspi\_deinit is shown as below:

**Table 3-800. Function qspi\_deinit**

Function name	qspi_deinit
Function prototype	void qspi_deinit(void);
Function descriptions	reset QSPI peripheral

<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset QSPI */
```

```
qspi_deinit();
```

### qspi\_struct\_para\_init

The description of qspi\_struct\_para\_init is shown as below:

**Table 3-801. Function qspi\_struct\_para\_init**

<b>Function name</b>	qspi_struct_para_init
<b>Function prototype</b>	void qspi_struct_para_init(qspi_init_struct *init_para);
<b>Function descriptions</b>	initialize the parameters of QSPI structure with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>init_para</b>	QSPI parameter structure, the structure members can refer to <a href="#">Table 3-797. Structure qspi_init_struct.</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of QSPI */
```

```
qspi_init_struct qspi_init_para;
```

```
qspi_struct_para_init(&qspi_init_para);
```

### qspi\_cmd\_struct\_para\_init

The description of qspi\_cmd\_struct\_para\_init is shown as below:

**Table 3-802. Function qspi\_cmd\_struct\_para\_init**

<b>Function name</b>	qspi_cmd_struct_para_init
<b>Function prototype</b>	void qspi_cmd_struct_para_init(qspi_command_struct *init_para);

<b>Function descriptions</b>	initialize the parameters of QSPI command structure with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>init_para</b>	QSPI command parameter structure, the structure members can refer to <a href="#">Table 3-798. Structure qspi_command_struct.</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of QSPI command structrue */
```

```
qspi_command_struct qspi_cmd_para;
```

```
qspi_cmd_struct_para_init(&qspi_cmd_para);
```

### qspi\_polling\_struct\_para\_init

The description of qspi\_polling\_struct\_para\_init is shown as below:

**Table 3-803. Function qspi\_polling\_struct\_para\_init**

<b>Function name</b>	qspi_polling_struct_para_init
<b>Function prototype</b>	void qspi_polling_struct_para_init(qspi_polling_struct *init_para);
<b>Function descriptions</b>	initialize the parameters of QSPI read polling structure with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>init_para</b>	QSPI read polling parameter structure, the structure members can refer to <a href="#">Table 3-799. Structure qspi_polling_struct.</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of QSPI read polling structrue */
```

```
qspi_polling_struct qspi_polling_para;
```

```
qspi_polling_struct_para_init(&qspi_polling_para);
```

## qspi\_init

The description of qspi\_init is shown as below:

**Table 3-804. Function qspi\_init**

<b>Function name</b>	qspi_init
<b>Function prototype</b>	void qspi_init(qspi_init_struct* qspi_struct);
<b>Function descriptions</b>	Initialize QSPI
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>qspi_struct</b>	QSPI parameter initialization structure, the structure members can refer to members of the structure <a href="#">Table 3-797. Structure qspi_init_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the init parameter structure */

qspi_struct_para_init(&qspi_init_para);

qspi_init_para.clock_mode = QSPI_CLOCK_MODE_0;

qspi_init_para.fifo_threshold = 4;

qspi_init_para.sample_shift = QSPI_SAMPLE_SHIFTING_HALFCYCLE;

qspi_init_para.cs_high_time = QSPI_CS_HIGH_TIME_2_CYCLE;

qspi_init_para.flash_size = 27;

qspi_init_para.prescaler = 4;

qspi_init(&qspi_init_para);
```

## qspi\_enable

The description of qspi\_enable is shown as below:

**Table 3-805. Function qspi\_enable**

<b>Function name</b>	qspi_enable
<b>Function prototype</b>	void qspi_enable(void);
<b>Function descriptions</b>	enable QSPI
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

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

Example:

```
/* enable QSPI */
qspi_enable();
```

### qspi\_disable

The description of qspi\_disable is shown as below:

**Table 3-806. Function qspi\_disable**

<b>Function name</b>	qspi_disable
<b>Function prototype</b>	void qspi_disable(void);
<b>Function descriptions</b>	disable QSPI
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable QSPI */
qspi_disable();
```

### qspi\_dma\_enable

The description of qspi\_dma\_enable is shown as below:

**Table 3-807. Function qspi\_dma\_enable**

<b>Function name</b>	qspi_dma_enable
<b>Function prototype</b>	void qspi_dma_enable(void);
<b>Function descriptions</b>	enable QSPI DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable QSPI DMA */
qspi_dma_enable();
```

### qspi\_dma\_disable

The description of qspi\_dma\_disable is shown as below:

**Table 3-808. Function qspi\_dma\_disable**

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

Example:

```
/* disable QSPI DMA */
qspi_dma_disable();
```

### qspi\_command\_config

The description of qspi\_command\_config is shown as below:

**Table 3-809. Function qspi\_command\_config**

Function name	qspi_command_config
Function prototype	void qspi_command_config(qspi_command_struct *cmd);
Function descriptions	configure QSPI command parameters
Precondition	-
The called functions	-
Input parameter{in}	
cmd	QSPI command parameter structure, the structure members can refer to <a href="#">Table 3-798. Structure qspi_command_struct</a> .



Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the QSPI command parameter structure */
qspi_cmd_struct_para_init(&qspi_cmd);

/* write enable */
qspi_cmd.instruction_mode = QSPI_INSTRUCTION_1_LINE;
qspi_cmd.instruction = WRITE_ENABLE_CMD;
qspi_cmd.addr_mode = QSPI_ADDR_NONE;
qspi_cmd.altebytes_mode = QSPI_ALTE_BYTES_NONE;
qspi_cmd.data_mode = QSPI_DATA_NONE;
qspi_cmd.dummycycles = 0;
qspi_cmd.sioo_mode = QSPI_SIOO_INST_EVERY_CMD;
qspi_command_config(&qspi_cmd);

```

### qspi\_polling\_config

The description of qspi\_polling\_config is shown as below:

**Table 3-810. Function qspi\_polling\_config**

Function name	qspi_polling_config
Function prototype	void qspi_polling_config(qspi_command_struct *cmd, qspi_polling_struct *cfg);
Function descriptions	configure QSPI read polling mode
Precondition	-
The called functions	-
Input parameter{in}	
cmd	QSPI command parameter structure, the structure members can refer to <a href="#">Table 3-798. Structure qspi_command_struct.</a>
Input parameter{in}	
cfg	QSPI read polling parameter structure, the structure members can refer to <a href="#">Table 3-799. Structure qspi_polling_struct.</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize the QSPI command parameter structure */

qspi_cmd_struct_para_init(&qspi_cmd);

/* initialize the QSPI read polling parameter structure */

qspi_polling_struct_para_init(&polling_cmd);

/* read enable */

polling_cmd.match = 0x02;

polling_cmd.mask = 0x02;

polling_cmd.match_mode = QSPI_MATCH_MODE_AND;

polling_cmd.statusbytes_size = 1;

polling_cmd.interval = 0x10;

polling_cmd.polling_stop = QSPI_POLLING_STOP_ENABLE;

qspi_cmd.instruction = READ_STATUS_REG1_CMD;

qspi_cmd.data_mode = QSPI_DATA_1_LINE;

qspi_polling_config(&qspi_cmd, &polling_cmd);

```

### qspi\_memorymapped\_config

The description of qspi\_memorymapped\_config is shown as below:

**Table 3-811. Function qspi\_memorymapped\_config**

Function name	qspi_memorymapped_config
Function prototype	void qspi_memorymapped_config(qspi_command_struct *cmd, uint16_t timeout, uint32_t toen);
Function descriptions	configure QSPI memory mapped mode
Precondition	-
The called functions	-
Input parameter{in}	
cmd	QSPI command parameter structure, the structure members can refer to <a href="#">Table 3-798. Structure qspi_command_struct</a> .
Input parameter{in}	
timeout	0-0xFFFF
Input parameter{in}	
toen	timeout counter
QSPI_TMOUT_DISABLE	disable timeout counter
QSPI_TMOUT_ENABLE	enable timeout counter

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

Example:

```

/* initialize the QSPI command parameter structure */

qspi_cmd_struct_para_init(&qspi_cmd);

qspi_cmd.instruction_mode = QSPI_INSTRUCTION_1_LINE;

qspi_cmd.instruction = RDID;

qspi_cmd.addr_mode = QSPI_ADDR_NONE;

qspi_cmd.addr_size = QSPI_ADDR_8_BITS;

qspi_cmd.addr = 0;

qspi_cmd.altebytes_mode = QSPI_ALTE_BYTES_1_LINE;

qspi_cmd.altebytes_size = QSPI_ALTE_BYTES_24_BITS;

qspi_cmd.altebytes = 0;

qspi_cmd.dummycycles = 0;

qspi_cmd.data_mode = QSPI_DATA_1_LINE;

qspi_cmd.data_length = 3;

qspi_cmd.sioo_mode = QSPI_SIOO_INST_EVERY_CMD;

qspi_memorymapped_config(&qspi_cmd, 0xf, QSPI_TMOUOUT_ENABLE);

```

### qspi\_data\_transmit

The description of qspi\_data\_transmit is shown as below:

**Table 3-812. Function qspi\_data\_transmit**

<b>Function name</b>	qspi_data_transmit
<b>Function prototype</b>	void qspi_data_transmit(uint8_t *tdata);
<b>Function descriptions</b>	QSPI transmit data function
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>tdata</b>	pointer to the data to be transmitted
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* QSPI transmit data */
uint8_t data[32];
qspi_data_transmit(&data);
```

### qspi\_data\_receive

The description of qspi\_data\_receive is shown as below:

**Table 3-813. Function qspi\_data\_receive**

Function name	qspi_data_receive
Function prototype	void qspi_data_transmit(uint8_t *tdata);
Function descriptions	QSPI transmit data function
Precondition	-
The called functions	-
Input parameter{in}	
rdata	pointer to the data to be received
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* QSPI receive data */
uint8_t data[32];
qspi_data_receive(&data);
```

### qspi\_transmission\_abort

The description of qspi\_transmission\_abort is shown as below:

**Table 3-814. Function qspi\_transmission\_abort**

Function name	qspi_transmission_abort
Function prototype	void qspi_transmission_abort(void);
Function descriptions	abort the current transmission
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* abort QSPI transmission */
qspi_transmission_abort();
```

### qspi\_output\_clock\_delay\_enable

The description of qspi\_output\_clock\_delay\_enable is shown as below:

**Table 3-815. Function qspi\_output\_clock\_delay\_enable**

<b>Function name</b>	qspi_output_clock_delay_enable
<b>Function prototype</b>	void qspi_output_clock_delay_enable(void);
<b>Function descriptions</b>	enable output clock delay
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable output clock delay */
qspi_output_clock_delay_enable();
```

### qspi\_output\_clock\_delay\_disable

The description of qspi\_output\_clock\_delay\_disable is shown as below:

**Table 3-816. Function qspi\_output\_clock\_delay\_disable**

<b>Function name</b>	qspi_output_clock_delay_disable
<b>Function prototype</b>	void qspi_output_clock_delay_disable (void);
<b>Function descriptions</b>	disable output clock delay
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* disable output clock delay */
```

```
qspi_output_clock_delay_disable();
```

### qspi\_output\_clock\_delay\_config

The description of qspi\_output\_clock\_delay\_config is shown as below:

**Table 3-817. Function qspi\_output\_clock\_delay\_config**

Function name	qspi_output_clock_delay_config
Function prototype	void qspi_output_clock_delay_config(uint32_t ck_delay);
Function descriptions	configure output clock delay
Precondition	-
The called functions	-
Input parameter{in}	
ck_delay	clock delay, 0~15
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure output clock delay */
```

```
qspi_output_clock_delay_config(1);
```

### qspi\_sample\_shift\_config

The description of qspi\_sample\_shift\_config is shown as below:

**Table 3-818. Function qspi\_sample\_shift\_config**

Function name	qspi_sample_shift_config
Function prototype	qspi_sample_shift_config(uint32_t sample_shift1, uint32_t sample_shift2);
Function descriptions	configure QSPI sample shift
Precondition	-
The called functions	-
Input parameter{in}	
sample_shift1	QSPI sample shift1
QSPI_SAMPLE_SHIFT1_NONE	no shift when sample data
QSPI_SAMPLE_SHIFT1_HALF_CYCLE	1/2 sck cycle shift

Input parameter{in}	
<b>sample_shift2</b>	QSPI sample shift2
<i>QSPI_SHIFTING_NONE</i>	no shift when sample data
<i>QSPI_SHIFTING_1_CYCLE</i>	1 sck cycle shift
<i>QSPI_SHIFTING_2_CYCLE</i>	2 sck cycle shift
<i>QSPI_SHIFTING_3_CYCLE</i>	3 sck cycle shift
<i>QSPI_SHIFTING_4_CYCLE</i>	4 sck cycle shift
<i>QSPI_SHIFTING_5_CYCLE</i>	5 sck cycle shift
<i>QSPI_SHIFTING_6_CYCLE</i>	6 sck cycle shift
<i>QSPI_SHIFTING_7_CYCLE</i>	7 sck cycle shift
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure QSPI sample shift */
```

```
qspi_sample_shift_config(QSPI_SAMPLE_SHIFTING_NONE, QSPI_SHIFTING_1_CYCLE);
```

### qspi\_receive\_clock\_sel

The description of qspi\_receive\_clock\_sel is shown as below:

**Table 3-819. Function qspi\_receive\_clock\_sel**

<b>Function name</b>	qspi_receive_clock_sel
<b>Function prototype</b>	void qspi_receive_clock_sel(uint32_t rcksel);
<b>Function descriptions</b>	select QSPI receive clock
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>rcksel</b>	receive clock select
<i>QSPI_RECEIVE_CLOCK_SCK</i>	QSPI receive clock choose SCK
<i>QSPI_RECEIVE_CLOCK_DQS</i>	QSPI receive clock choose DQS

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select QSPI receive clock */
qspi_receive_clock_sel(QSPI_RECEIVE_CLOCK_DQS);
```

### qspi\_delay\_scan\_enable

The description of qspi\_delay\_scan\_enable is shown as below:

**Table 3-820. Function qspi\_delay\_scan\_enable**

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

Example:

```
/* enable delay scan */
void qspi_delay_scan_enable();
```

### qspi\_delay\_scan\_disable

The description of qspi\_delay\_scan\_disable is shown as below:

**Table 3-821. Function qspi\_delay\_scan\_disable**

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



-	-
Return value	
-	-

Example:

```
/* disable delay scan */
```

```
void qspi_delay_scan_disable();
```

### qspi\_csn\_edge\_cycle

The description of qspi\_csn\_edge\_cycle is shown as below:

**Table 3-822. Function qspi\_csn\_edge\_cycle**

<b>Function name</b>	qspi_csn_edge_cycle
<b>Function prototype</b>	void qspi_csn_edge_cycle(uint32_t csn_cycle);
<b>Function descriptions</b>	select csn falls and rises 1 or 2 sck cycles
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>csn_cycle</b>	csn clock cycle
QSPI_CSN_1_CYCLE	csn falls and rises 1 sck cycle
QSPI_CSN_2_CYCLE	csn falls and rises 2 sck cycle
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select csn falls and rises 2 sck cycles */
```

```
qspi_csn_edge_cycle(QSPI_CSN_2_CYCLE);
```

### qspi\_flag\_get

The description of qspi\_flag\_get is shown as below:

**Table 3-823. Function qspi\_flag\_get**

<b>Function name</b>	qspi_flag_get
<b>Function prototype</b>	FlagStatus qspi_flag_get(uint32_t flag);
<b>Function descriptions</b>	get QSPI flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	QSPI flag

QSPI_FLAG_BUSY	busy flag
QSPI_FLAG_TERR	transfer error flag
QSPI_FLAG_TC	transfer complete flag
QSPI_FLAG_FT	FIFO threshold flag
QSPI_FLAG_RPMF	read polling match flag
QSPI_FLAG_TMOUT	timeout flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get QSPI transfer complete flag */
```

```
FlagStatus status;
```

```
status = qspi_flag_get(QSPI_FLAG_TC);
```

### qspi\_flag\_clear

The description of qspi\_flag\_clear is shown as below:

**Table 3-824. Function qspi\_flag\_clear**

Function name	qspi_flag_clear
Function prototype	void qspi_flag_clear(uint32_t flag);
Function descriptions	clear QSPI flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	QSPI flag
QSPI_FLAG_TERR	transfer error flag
QSPI_FLAG_TC	transfer complete flag
QSPI_FLAG_RPMF	read polling match flag
QSPI_FLAG_TMOUT	timeout flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear QSPI transfer complete flag status */
```

```
qspi_flag_clear(QSPI_FLAG_TC);
```

## qspi\_interrupt\_enable

The description of qspi\_interrupt\_enable is shown as below:

**Table 3-825. Function qspi\_interrupt\_enable**

<b>Function name</b>	qspi_interrupt_enable
<b>Function prototype</b>	void qspi_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable QSPI interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	QSPI interrupt
QSPI_INT_TC	transfer complete interrupt
QSPI_INT_FT	FIFO threshold interrupt
QSPI_INT_TERR	transfer error interrupt
QSPI_INT_RPMF	read polling match interrupt
QSPI_INT_TMOUT	timeout interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

Example:

```
/* enable QSPI transfer complete interrupt */
```

```
qspi_interrupt_enable(QSPI_INT_TC);
```

## qspi\_interrupt\_disable

The description of qspi\_interrupt\_disable is shown as below:

**Table 3-826. Function qspi\_interrupt\_disable**

<b>Function name</b>	qspi_interrupt_disable
<b>Function prototype</b>	void qspi_interrupt_disable( uint8_t interrupt);
<b>Function descriptions</b>	disable QSPI interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	QSPI interrupt
QSPI_INT_TC	transfer complete interrupt
QSPI_INT_FT	FIFO threshold interrupt
QSPI_INT_TERR	transfer error interrupt
QSPI_INT_RPMF	read polling match interrupt
QSPI_INT_TMOUT	timeout interrupt
<b>Output parameter{out}</b>	

-	-
Return value	

Example:

```
/* disable QSPI transfer complete interrupt */
```

```
qspi_interrupt_disable(QSPI_INT_TC);
```

### qspi\_interrupt\_flag\_get

The description of qspi\_interrupt\_flag\_get is shown as below:

**Table 3-827. Function qspi\_interrupt\_flag\_get**

<b>Function name</b>	qspi_interrupt_flag_get
<b>Function prototype</b>	FlagStatus qspi_interrupt_flag_get(uint32_t int_flag);
<b>Function descriptions</b>	get QSPI interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	QSPI interrupt flag
QSPI_INT_FLAG_TRANSFER	transfer error interrupt flag
QSPI_INT_FLAG_TC	transfer complete interrupt flag
QSPI_INT_FLAG_FT	FIFO threshold interrupt flag
QSPI_INT_FLAG_RPMF	read polling match interrupt flag
QSPI_INT_FLAG_TMO UT	timeout interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get QSPI transfer complete flag */
```

```
FlagStatus status;
```

```
status = qspi_flag_get(QSPI_FLAG_TC);
```

### qspi\_interrupt\_flag\_clear

The description of qspi\_interrupt\_flag\_clear is shown as below:

Table 3-828. Function `qspi_interrupt_flag_clear`

<b>Function name</b>	<code>qspi_interrupt_flag_clear</code>
<b>Function prototype</b>	<code>void qspi_interrupt_flag_clear(uint32_t int_flag);</code>
<b>Function descriptions</b>	clear QSPI interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	QSPI interrupt flag
<code>QSPI_INT_FLAG_TER</code> <i>R</i>	transfer error interrupt flag
<code>QSPI_INT_FLAG_TC</code>	transfer complete interrupt flag
<code>QSPI_INT_FLAG_RPM</code> <i>F</i>	read polling match interrupt flag
<code>QSPI_INT_FLAG_TMO</code> <i>UT</i>	timeout interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear QSPI transfer complete flag status */
qspi_flag_clear(QSPI_FLAG_TC);
```

## 3.25. 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 [3.25.1](#), the RCU firmware functions are introduced in chapter [3.25.2](#).

### 3.25.1. Descriptions of Peripheral registers

RCU registers are listed in the table shown as below:

Table 3-829. RCU Registers

Registers	Descriptions
RCU_CTL	control register
RCU_PLL	PLL register
RCU_CFG0	clock configuration register 0
RCU_INT	clock interrupt register
RCU_AHB1RST	AHB1 reset register

Registers	Descriptions
RCU_AHB2RST	AHB2 reset register
RCU_AHB3RST	AHB3 reset register
RCU_APB1RST	APB1 reset register
RCU_APB2RST	APB2 reset register
RCU_APB3RST	APB3 reset register
RCU_AHB1EN	AHB1 enable register
RCU_AHB2EN	AHB2 enable register
RCU_AHB3EN	AHB3 enable register
RCU_APB1EN	APB1 enable register
RCU_APB2EN	APB2 enable register
RCU_APB3EN	APB3 enable register
RCU_AHB1SPDPE N	AHB1 sleep mode and deepsleep mode enable register
RCU_AHB2SPDPE N	AHB2 sleep mode and deepsleep mode enable register
RCU_AHB3SPDPE N	AHB3 sleep mode and deepsleep mode enable register
RCU_APB1SPDPE N	APB1 sleep mode and deepsleep mode enable register
RCU_APB2 SPDPEN	APB2 sleep mode and deepsleep mode enable register
RCU_APB3 SPDPEN	APB3 sleep mode and deepsleep mode enable register
RCU_BDCTL	backup domain control register
RCU_RSTSCK	reset source / clock register
RCU_CFG1	clock configuration register 1
RCU_CFG2	clock configuration register 2

### 3.25.2. Descriptions of Peripheral functions

**Table 3-830. RCU firmware function**

Function name	Function description
rcu_deinit	deinitialize the RCU
rcu_periph_clock_enable	enable the peripherals clock
rcu_periph_clock_disable	disable the peripherals clock
rcu_periph_clock_sleep_enable	enable the peripherals clock when sleep mode
rcu_periph_clock_sleep_disable	disable the peripherals clock when sleep mode
rcu_periph_reset_enable	reset the peripherals
rcu_periph_reset_disable	disable reset the peripheral
rcu_bkp_reset_enable	enable the BKP domain reset
rcu_bkp_reset_disable	disable the BKP domain reset

Function name	Function description
rcu_system_clock_source_config	configure the system clock source
rcu_system_clock_source_get	get the system clock source
rcu_ahb_clock_config	configure the AHB clock prescaler selection
rcu_apb1_clock_config	configure the APB1 clock prescaler selection
rcu_apb2_clock_config	configure the APB2 clock prescaler selection
rcu_apb3_clock_config	configure the APB3 clock prescaler selection
rcu_ckout_config	configure the CK_OUT clock source and divider
rcu_lsckout_enable	enable the low speed clock output
rcu_lsckout_disable	disable the low speed clock output
rcu_lsckout_config	configure the LCKOUT clock source
rcu_pll_source_config	configure the PLLs clock source selection
rcu_pll_config	configure the PLL
rcu_pll_clock_output_enable	enable the pllq pllq pllr divider output
rcu_pll_clock_output_disable	disable the pllq pllq pllr divider output
rcu_rtc_clock_config	configure the RTC clock source selection
rcu_usart_clock_config	configure the USARTx(x=0,1,2) clock source selection
rcu_i2c_clock_config	configure the I2Cx(x=0,1,2,3) clock source selection
rcu_can_clock_config	configure the CANx(x=0,1,2) clock source selection
rcu_adc_clock_config	configure the ADCx(x=0,1,2) clock source selection
rcu_hpdf_clock_config	configure the HPDF clock source selection
rcu_hpdfaudio_clock_config	configure the HPDF AUDIO clock source selection
rcu_trng_clock_config	configure the TRNG prescaler selection
rcu_lptimer_clock_config	configure the LPTIMER clock source selection
rcu_qspi_clock_config	configure the QSPI clock source selection
rcu_hrtimer_clock_config	configure the HRTIMER 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, HXTALEN or LXTALEN must be reset before it
rcu_osci_bypass_mode_disable	disable the oscillator bypass mode, HXTALEN or LXTALEN must be reset before it
rcu_osci_stablization_rest	oscillator stabilization reset
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_clock_freq_get	get the system clock, bus and peripheral clock frequency

Function name	Function description
rcu_flag_get	get the clock stabilization and peripheral reset flags
rcu_all_reset_flag_clear	clear the reset flag
rcu_interrupt_enable	enable the stabilization interrupt
rcu_interrupt_disable	disable the stabilization interrupt
rcu_interrupt_flag_get	get the clock stabilization interrupt and ckm flags
rcu_interrupt_flag_clear	clear the interrupt flags

## Enum rcu\_periph\_enum

**Table 3-831. Enum rcu\_periph\_enum**

enum name	Function description
RCU_CRC	CRC clock
RCU_CLA	CLA clock
RCU_DMA0	DMA0 clock
RCU_DMA1	DMA1 clock
RCU_DMAMUX	DMAMUX clock
RCU_FFT	FFT clock
RCU_FAC	FAC clock
RCU_CAU	CAU clock
RCU_TRNG	TRNG clock
RCU_TMU	TMU 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_GPIOG	GPIOG clock
RCU_EXMC	EXMC clock
RCU_QSPI	QSPI clock
RCU_TIMER1	TIMER1 clock
RCU_TIMER2	TIMER2 clock
RCU_TIMER3	TIMER3 clock
RCU_TIMER4	TIMER4 clock
RCU_TIMER5	TIMER5 clock
RCU_TIMER6	TIMER6 clock
RCU_LPTIMER	LPTIMER clock
RCU_WWDGT	TIMER51 clock
RCU_SPI1	SPI1 clock
RCU_SPI2	SPI2 clock
RCU_USART1	USART1 clock
RCU_USART2	USART2 clock



enum name	Function description
RCU_UART3	UART3 clock
RCU_UART4	UART4 clock
RCU_I2C0	I2C0 clock
RCU_I2C1	I2C1 clock
RCU_I2C2	I2C2 clock
RCU_I2C3	I2C3 clock
RCU_PMU	PMU clock
RCU_TIMER0	TIMER0 clock
RCU_TIMER7	TIMER7 clock
RCU_VREF	VREF clock
RCU_CMP	CMP clock
RCU_USART0	USART0 clock
RCU_CAN0	CAN0 clock
RCU_CAN1	CAN1 clock
RCU_CAN2	CAN2 clock
RCU_SPI0	SPI0 clock
RCU_SYSCFG	SYSCFG clock
RCU_TIMER19	TIMER19 clock
RCU_TIMER14	TIMER14 clock
RCU_TIMER15	TIMER15 clock
RCU_TIMER16	TIMER16 clock
RCU_HPDF	HPDF clock
RCU_HRTIMER	HRTIMER clock
RCU_TRIGSEL	TRIGSEL clock
RCU_ADC0	ADC0 clock
RCU_ADC1	ADC1 clock
RCU_ADC2	ADC2 clock
RCU_ADC3	ADC3 clock
RCU_DACHOLD	DACHOLD clock
RCU_DAC0	DAC0 clock
RCU_DAC1	DAC1 clock
RCU_DAC2	DAC2 clock
RCU_DAC3	DAC3 clock
RCU_RTC	RTC clock

### Enum rcu\_periph\_sleep\_enum

Table 3-832. Enum rcu\_periph\_sleep\_enum

enum name	Function description
RCU_FMC_SLP	FMC clock
RCU_CRC_SLP	CRC clock
RCU_CLA_SLP	CLA clock

enum name	Function description
RCU_SRAM0_SLP	SRAM0 clock
RCU_SRAM1_SLP	SRAM1 clock
RCU_TCMRAM_SLP	TCMSRAM clock
RCU_DMA0_SLP	DMA0 clock
RCU_DMA1_SLP	DMA1 clock
RCU_DMAMUX_SLP	DMAMUX clock
RCU_FFT_SLP	FFT clock
RCU_FAC_SLP	FAC clock
RCU_CAU_SLP	CAU clock
RCU_TRNG_SLP	TRNG clock
RCU_TMU_SLP	TMU clock
RCU_GPIOA_SLP	GPIOA clock
RCU_GPIOB_SLP	GPIOB clock
RCU_GPIOC_SLP	GPIOC clock
RCU_GPIOD_SLP	GPIOD clock
RCU_GPIOE_SLP	GPIOE clock
RCU_GPIOF_SLP	GPIOF clock
RCU_GPIOG_SLP	GPIOG clock
RCU_EXMC_SLP	EXMC clock
RCU_QSPI_SLP	QSPI clock
RCU_TIMER1_SLP	TIMER1 clock
RCU_TIMER2_SLP	TIMER2 clock
RCU_TIMER3_SLP	TIMER3 clock
RCU_TIMER4_SLP	TIMER4 clock
RCU_TIMER5_SLP	TIMER5 clock
RCU_TIMER6_SLP	TIMER6 clock
RCU_LPTIMER_SLP	LPTIMER clock
RCU_WWDGT_SLP	TIMER51 clock
RCU_SPI1_SLP	SPI1 clock
RCU_SPI2_SLP	SPI2 clock
RCU_USART1_SLP	USART1 clock
RCU_USART2_SLP	USART2 clock
RCU_UART3_SLP	UART3 clock
RCU_UART4_SLP	UART4 clock
RCU_I2C0_SLP	I2C0 clock
RCU_I2C1_SLP	I2C1 clock
RCU_I2C2_SLP	I2C2 clock
RCU_I2C3_SLP	I2C3 clock
RCU_PMU_SLP	PMU clock
RCU_TIMER0_SLP	TIMER0 clock
RCU_TIMER7_SLP	TIMER7 clock

enum name	Function description
RCU_VREF_SLP	VREF clock
RCU_CMP_SLP	CMP clock
RCU_USART0_SLP	USART0 clock
RCU_CAN0_SLP	CAN0 clock
RCU_CAN1_SLP	CAN1 clock
RCU_CAN2_SLP	CAN2 clock
RCU_SPI0_SLP	SPI0 clock
RCU_SYSCFG_SLP	SYSCFG clock
RCU_TIMER19_SLP	TIMER19 clock
RCU_TIMER14_SLP	TIMER14 clock
RCU_TIMER15_SLP	TIMER15 clock
RCU_TIMER16_SLP	TIMER16 clock
RCU_HPDF_SLP	HPDF clock
RCU_HRTIMER_SLP	HRTIMER clock
RCU_TRIGSEL_SLP	TRIGSEL clock
RCU_ADC0_SLP	ADC0 clock
RCU_ADC1_SLP	ADC1 clock
RCU_ADC2_SLP	ADC2 clock
RCU_ADC3_SLP	ADC3 clock
RCU_DACHOLD_SLP	DACHOLD clock
RCU_DAC0_SLP	DAC0 clock
RCU_DAC1_SLP	DAC1 clock
RCU_DAC2_SLP	DAC2 clock
RCU_DAC3_SLP	DAC3 clock

### Enum rcu\_periph\_reset\_enum

**Table 3-833. Enum rcu\_periph\_reset\_enum**

enum name	Function description
RCU_CRCRST	reset CRC
RCU_CLARST	reset CLA
RCU_DMA0RST	reset DMA0
RCU_DMA1RST	reset DMA1
RCU_DMAMUXRST	reset DMAMUX
RCU_FFTRST	reset FFT
RCU_FACRST	reset FAC
RCU_CAURST	reset CAU
RCU_TRNGRST	reset TRNG
RCU_TMURST	reset TMU
RCU_GPIOARST	reset GPIOA
RCU_GPIOBRST	reset GPIOB
RCU_GPIOCRST	reset GPIOC

enum name	Function description
RCU_GPIODRST	reset GPIOD
RCU_GPIOERST	reset GPIOE
RCU_GPIOFRST	reset GPIOF
RCU_GPIOGRST	reset GPIOG
RCU_EXMCRST	reset EXMC
RCU_QSPIRST	reset QSPI
RCU_TIMER1RST	reset TIMER1
RCU_TIMER2RST	reset TIMER2
RCU_TIMER3RST	reset TIMER3
RCU_TIMER4RST	reset TIMER4
RCU_TIMER5RST	reset TIMER5
RCU_TIMER6RST	reset TIMER6
RCU_LPTIMERRST	reset LPTIMER
RCU_WWDGTRST	reset TIMER51
RCU_SPI1RST	reset SPI1
RCU_SPI2RST	reset SPI2
RCU_USART1RST	reset USART1
RCU_USART2RST	reset USART2
RCU_UART3RST	reset UART3
RCU_UART4RST	reset UART4
RCU_I2C0RST	reset I2C0
RCU_I2C1RST	reset I2C1
RCU_I2C2RST	reset I2C2
RCU_I2C3RST	reset I2C3
RCU_PMURST	reset PMU
RCU_TIMER0RST	reset TIMER0
RCU_TIMER7RST	reset TIMER7
RCU_VREFRST	reset VREF
RCU_CMPRST	reset CMP
RCU_USART0RST	reset USART0
RCU_CAN0RST	reset CAN0
RCU_CAN1RST	reset CAN1
RCU_CAN2RST	reset CAN2
RCU_SPI0RST	reset SPI0
RCU_SYSCFGRST	reset SYSCFG
RCU_TIMER19RST	reset TIMER19
RCU_TIMER14RST	reset TIMER14
RCU_TIMER15RST	reset TIMER15
RCU_TIMER16RST	reset TIMER16
RCU_HPDRST	reset HPDF
RCU_HRTIMERRST	reset HRTIMER

enum name	Function description
RCU_TRIGSELRST	reset TRIGSEL
RCU_ADC0RST	reset ADC0
RCU_ADC1RST	reset ADC1
RCU_ADC2RST	reset ADC2
RCU_ADC3RST	reset ADC3
RCU_DACHOLDRST	reset DACHOLD
RCU_DAC0RST	reset DAC0
RCU_DAC1RST	reset DAC1
RCU_DAC2RST	reset DAC2
RCU_DAC3RST	reset DAC3

### Enum rcu\_flag\_enum

**Table 3-834. Enum rcu\_flag\_enum**

enum name	Function description
RCU_FLAG_IRC8MST B	IRC8M stabilization flags
RCU_FLAG_HXTALST B	HXTAL stabilization flags
RCU_FLAG_PLLSTB	PLL stabilization flags
RCU_FLAG_LXTALST B	LXTAL stabilization flags
RCU_FLAG_IRC32KST B	IRC32K stabilization flags
RCU_FLAG_OBLRST	OBL reset flags
RCU_FLAG_BORRST	BOR reset flags
RCU_FLAG_EPRST	external PIN reset flags
RCU_FLAG_PORRST	power reset flags
RCU_FLAG_SWRST	software reset flags
RCU_FLAG_FWDGTR ST	FWDGT reset flags
RCU_FLAG_WWDGTR ST	WWDGT reset flags
RCU_FLAG_LPRST	low-power reset flags

### Enum rcu\_int\_flag\_enum

**Table 3-835. Enum rcu\_int\_flag\_enum**

enum name	Function description
RCU_INT_FLAG_IRC3 2KSTB	IRC32K stabilization interrupt flag
RCU_INT_FLAG_LXTA	LXTAL stabilization interrupt flag

enum name	Function description
LSTB	
RCU_INT_FLAG_IRC8 MSTB	IRC8M stabilization interrupt flag
RCU_INT_FLAG_HXTA LSTB	HXTAL stabilization interrupt flag
RCU_INT_FLAG_PLLS TB	PLL stabilization interrupt flag
RCU_INT_FLAG_CKM	HXTAL clock stuck interrupt flag
RCU_INT_FLAG_LCK M	LXTAL clock stuck interrupt flag

### Enum rcu\_int\_flag\_clear\_enum

**Table 3-836. Enum rcu\_int\_flag\_clear\_enum**

enum name	Function description
RCU_INT_FLAG_IRC3 2KSTB_CLR	IRC32K stabilization interrupt flags clear
RCU_INT_FLAG_LXTA LSTB_CLR	LXTAL stabilization interrupt flags clear
RCU_INT_FLAG_IRC8 MSTB_CLR	IRC8M stabilization interrupt flags clear
RCU_INT_FLAG_HXTA LSTB_CLR	HXTAL stabilization interrupt flags clear
RCU_INT_FLAG_PLLS TB_CLR	PLL stabilization interrupt flags clear
RCU_INT_FLAG_CKM _CLR	CKM interrupt flags clear
RCU_INT_FLAG_LCK M_CLR	LCKM interrupt flags clear

### Enum rcu\_int\_enum

**Table 3-837. Enum rcu\_int\_enum**

enum name	Function description
RCU_INT_IRC32KSTB	IRC32K 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

## Enum rcu\_osci\_type\_enum

**Table 3-838. Enum rcu\_osci\_type\_enum**

enum name	Function description
RCU_HXTAL	HXTAL
RCU_LXTAL	LXTAL
RCU_IRC8M	IRC8M
RCU_IRC32K	IRC32K
RCU_PLL_CK	PLL

## Enum rcu\_clock\_freq\_enum

**Table 3-839. 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_APB3	APB3 clock
CK_PLLP	PLLP clock
CK_PLLR	PLLR clock
CK_PLLQ	PLLQ clock
CK_USART0	USART0 clock
CK_USART1	USART1 clock
CK_USART2	USART2 clock

## Enum usart\_idx\_enum

**Table 3-840. Enum usart\_idx\_enum**

enum name	Function description
IDX_USART0	idnex of USART0
IDX_USART1	idnex of USART1
IDX_USART2	idnex of USART2

## Enum i2c\_idx\_enum

**Table 3-841. Enum i2c\_idx\_enum**

enum name	Function description
IDX_I2C0	idnex of I2C0
IDX_I2C1	idnex of I2C1
IDX_I2C2	idnex of I2C2
IDX_I2C3	idnex of I2C3

## Enum can\_idx\_enum

**Table 3-842. Enum can\_idx\_enum**

enum name	Function description
IDX_CAN0	idnex of CAN0
IDX_CAN1	idnex of CAN1
IDX_CAN2	idnex of CAN2

## Enum adc\_idx\_enum

**Table 3-843. Enum adc\_idx\_enum**

enum name	Function description
IDX_ADC0	idnex of ADC0
IDX_ADC1	idnex of ADC1
IDX_ADC2	idnex of ADC2
IDX_ADC3	idnex of ADC3

## rcu\_deinit

The description of rcu\_deinit is shown as below:

**Table 3-844. Function rcu\_deinit**

<b>Function name</b>	rcu_deinit
<b>Function prototype</b>	void rcu_deinit(void);
<b>Function descriptions</b>	deinitialize the RCU
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset RCU */
rcu_deinit();
```

## rcu\_periph\_clock\_enable

The description of rcu\_periph\_clock\_enable is shown as below:

**Table 3-845. Function rcu\_periph\_clock\_enable**

<b>Function name</b>	rcu_periph_clock_enable
<b>Function prototype</b>	void rcu_periph_clock_enable(rcu_periph_enum periph);



<b>Function descriptions</b>	enable the peripherals clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>periph</b>	RCU peripherals, refer to <a href="#">Table 3-831. Enum rcu_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-846. Function rcu\_periph\_clock\_disable**

<b>Function name</b>	rcu_periph_clock_disable
<b>Function prototype</b>	void rcu_periph_clock_disable(rcu_periph_enum periph);
<b>Function descriptions</b>	disable the peripherals clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>periph</b>	RCU peripherals, refer to <a href="#">Table 3-831. Enum rcu_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### rcu\_periph\_clock\_sleep\_enable

The description of rcu\_periph\_clock\_sleep\_enable is shown as below:

**Table 3-847. Function rcu\_periph\_clock\_sleep\_enable**

<b>Function name</b>	rcu_periph_clock_sleep_enable
<b>Function prototype</b>	void rcu_periph_clock_sleep_enable(rcu_periph_sleep_enum periph);
<b>Function descriptions</b>	enable the peripherals clock when in sleep mode

Precondition	-
The called functions	-
Input parameter{in}	
periph	RCU peripherals, refer to <a href="#">Table 3-832. Enum rcu_periph_sleep_enum</a>
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 3-848. 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 <a href="#">Table 3-832. Enum rcu_periph_sleep_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the FMC clock when in sleep mode */
```

```
rcu_periph_clock_sleep_disable(RCU_FMC_SLP);
```

### rcu\_periph\_reset\_enable

The description of rcu\_periph\_reset\_enable is shown as below:

**Table 3-849. 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 <a href="#">Table 3-833. Enum rcu_periph_reset_enum</a>
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 3-850. 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 <a href="#">Table 3-833. Enum rcu_periph_reset_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### rcu\_bkp\_reset\_enable

The description of rcu\_bkp\_reset\_enable is shown as below:

**Table 3-851. 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 3-852. 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 3-853. 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	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_sys</b>	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_PLLP</i>	select CK_PLLP as the CK_SYS source
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-854. Function rcu\_system\_clock\_source\_get**

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

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 3-855. Function rcu\_ahb\_clock\_config

<b>Function name</b>	rcu_ahb_clock_config
<b>Function prototype</b>	void rcu_ahb_clock_config(uint32_t ck_ahb);
<b>Function descriptions</b>	configure the AHB clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_ahb</b>	AHB clock prescaler selection
<i>RCU_AHB_CKSYS_DIVx</i>	select CK_SYS / x, (x = 1, 2, 4, 8, 16, 64, 128, 256, 512)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-856. Function rcu\_apb1\_clock\_config

<b>Function name</b>	rcu_apb1_clock_config
<b>Function prototype</b>	void rcu_apb1_clock_config(uint32_t ck_apb1);
<b>Function descriptions</b>	configure the APB1 clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_apb1</b>	APB1 clock prescaler selection
<i>RCU_APB1_CKAHB_DIVx</i>	select (CK_AHB / x) as CK_APB1 (x = 1, 2, 4, 8, 16)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-857. Function rcu\_apb2\_clock\_config**

<b>Function name</b>	rcu_apb2_clock_config
<b>Function prototype</b>	void rcu_apb2_clock_config(uint32_t ck_apb2);
<b>Function descriptions</b>	configure the APB2 clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_apb2</b>	APB2 clock prescaler selection
<i>RCU_APB2_CK_AHB_D</i> <i>IVx</i>	select (CK_AHB / x) as CK_APB2 clock (x = 1, 2, 4, 8, 16)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB2 */
```

```
rcu_apb2_clock_config(RCU_APB2_CK_AHB_DIV8);
```

## rcu\_apb3\_clock\_config

The description of rcu\_apb3\_clock\_config is shown as below:

**Table 3-858. Function rcu\_apb3\_clock\_config**

<b>Function name</b>	rcu_apb3_clock_config
<b>Function prototype</b>	void rcu_apb3_clock_config(uint32_t ck_apb3);
<b>Function descriptions</b>	configure the APB3 clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_apb3</b>	APB3 clock prescaler selection
<i>RCU_APB3_CK_AHB_D</i> <i>IVx</i>	select (CK_AHB / x) as CK_APB3 clock (x = 1, 2, 4, 8, 16)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB3 */
```

```
rcu_apb3_clock_config(RCU_APB3_CK_AHB_DIV8);
```

## rcu\_ckout\_config

The description of rcu\_ckout\_config is shown as below:

**Table 3-859. Function rcu\_ckout\_config**

<b>Function name</b>	rcu_ckout_config
<b>Function prototype</b>	void rcu_ckout_config(uint32_t ckout_src, uint32_t ckout_div);
<b>Function descriptions</b>	configure the CK_OUT clock source and divider
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ckout_src</b>	CK_OUT clock source selection
RCU_CKOUTSRC_IRC32K	IRC64M selected
RCU_CKOUT0SRC_LXTAL	LXTAL selected
RCU_CKOUTSRC_HXTAL	HXTAL selected
RCU_CKOUTSRC_IRC8M	PLL0P selected
RCU_CKOUTSRC_HXTAL	IRC48M selected
RCU_CKOUTSRC_PLLP	PER selected
<b>Output parameter{out}</b>	
<b>ckout_div</b>	CK_OUT divider
RCU_CKOUT_DIVx	CK_OUT is divided by x(x = 1, 2, 4, 8, 16)
<b>Return value</b>	
-	-

Example:

```
/* configure the HXTAL as CK_OUT clock source */
```

```
rcu_ckout_config(RCU_CKOUTSRC_HXTAL, RCU_CKOUT_DIV1);
```

## rcu\_lsckout\_enable

The description of rcu\_lsckout\_enable is shown as below:

**Table 3-860. Function rcu\_lsckout\_enable**

<b>Function name</b>	rcu_lsckout_enable
----------------------	--------------------



<b>Function prototype</b>	void rcu_lsckout_enable(void);
<b>Function descriptions</b>	enable the low speed clock output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the low speed clock output */
rcu_lsckout_enable ();
```

### rcu\_lsckout\_disable

The description of rcu\_lsckout\_disable is shown as below:

**Table 3-861. Function rcu\_lsckout\_disable**

<b>Function name</b>	rcu_lsckout_disable
<b>Function prototype</b>	void rcu_lsckout_disable(void);
<b>Function descriptions</b>	disable the low speed clock output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the low speed clock output */
rcu_lsckout_disable ();
```

### rcu\_lsckout\_config

The description of rcu\_lsckout\_config is shown as below:

**Table 3-862. Function rcu\_lsckout\_config**

<b>Function name</b>	rcu_lsckout_config
<b>Function prototype</b>	void rcu_lsckout_config(uint32_t ls_ckout_src);

<b>Function descriptions</b>	configure the LSCKOUT clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ls_ckout_src</b>	LSCKOUT clock source selection
<i>RCU_LSCKOUTSRC_IRC32K</i>	IRC32K selected
<i>RCU_LSCKOUTSRC_LXTAL</i>	LXTAL selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the IRC32K as LSCK_OUT clock source */
rcu_lsckout_config(RCU_LSCKOUTSRC_IRC32K);
```

### rcu\_pll\_source\_config

The description of rcu\_pll\_source\_config is shown as below:

**Table 3-863. Function rcu\_pll\_source\_config**

<b>Function name</b>	rcu_pll_source_config
<b>Function prototype</b>	void rcu_pll_source_config(uint32_t pll_src);
<b>Function descriptions</b>	configure PLL clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll_src</b>	PLL clock source selection
<i>RCU_PLLSRC_IRC8M</i>	select IRC8M as PLL source clock
<i>RCU_PLLSRC_HXTAL</i>	select HXTAL as PLL source clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select RCU_PLLSRC_IRC8M as the PLL clock source */
rcu_pll_source_config(RCU_PLLSRC_IRC8M);
```

## rcu\_pll\_config

The description of rcu\_pll\_config is shown as below:

**Table 3-864. Function rcu\_pll\_config**

<b>Function name</b>	rcu_pll_config
<b>Function prototype</b>	ErrStatus rcu_pll_config(uint32_t pll_psc, uint32_t pll_n, uint32_t pll_p, uint32_t pll_q, uint32_t pll_r);
<b>Function descriptions</b>	configure the main PLL clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll_psc</b>	the PLL VCO source clock prescaler
<i>uint32_t</i>	1-16
<b>Input parameter{in}</b>	
<b>pll_n</b>	the PLL VCO clock multi factor
<i>uint32_t</i>	8-180
<b>Input parameter{in}</b>	
<b>pll_p</b>	the PLLP output frequency division factor from PLL VCO clock
<i>uint32_t</i>	2,4,6,8
<b>Input parameter{in}</b>	
<b>pll_q</b>	the PLLQ output frequency division factor from PLL VCO clock
<i>uint32_t</i>	2-15
<b>Input parameter{in}</b>	
<b>pll_r</b>	the PLLR output frequency division factor from PLL VCO clock
<i>uint32_t</i>	2-31
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR-

Example:

```
/* configure the PLL */
```

```
rcu_pll_config(4, 90, 1, 2, 2);
```

## rcu\_pll\_clock\_output\_enable

The description of rcu\_pll\_clock\_output\_enable is shown as below:

**Table 3-865. Function rcu\_pll\_clock\_output\_enable**

<b>Function name</b>	rcu_pll_clock_output_enable
<b>Function prototype</b>	void rcu_pll_clock_output_enable(uint32_t pll_x);
<b>Function descriptions</b>	enable the pll_p pll_q pll_r divider output
<b>Precondition</b>	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>plx</b>	the output pll enable
<i>RCU_PLLP</i>	PLLp divider output enable
<i>RCU_PLLQ</i>	PLLQ divider output enable
<i>RCU_PLLR</i>	PLLR divider output enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable PLLP divider output */
```

```
rcu_pll_clock_output_enable(RCU_PLLP);
```

### rcu\_pll\_clock\_output\_disable

The description of rcu\_pll\_clock\_output\_disable is shown as below:

**Table 3-866. Function rcu\_pll\_clock\_output\_disable**

<b>Function name</b>	rcu_pll_clock_output_disable
<b>Function prototype</b>	void rcu_pll_clock_output_disable(uint32_t pllx);
<b>Function descriptions</b>	disable the pll p pll q pll r divider output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>plx</b>	the output pll disable
<i>RCU_PLLP</i>	PLLp divider output disable
<i>RCU_PLLQ</i>	PLLQ divider output disable
<i>RCU_PLLR</i>	PLLR divider output disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable PLLP divider output */
```

```
rcu_pll_clock_output_disable(RCU_PLLP);
```

### rcu\_rtc\_clock\_config

The description of rcu\_rtc\_clock\_config is shown as below:

**Table 3-867. Function rcu\_rtc\_clock\_config**

<b>Function name</b>	rcu_rtc_clock_config
<b>Function prototype</b>	void rcu_rtc_clock_config(uint32_t rtc_clock_source);
<b>Function descriptions</b>	configure the RTC clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_clock_source</b>	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_IRC32K</i>	CK_IRC32K selected as RTC source clock
<i>RCU_RTCSRC_HXTAL_DIV32</i>	CK_HXTAL /32 selected as RTC source clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the RTC clock source selection */
```

```
rcu_rtc_clock_config(RCU_RTCSRC_IRC32K);
```

### rcu\_usart\_clock\_config

The description of rcu\_usart\_clock\_config is shown as below:

**Table 3-868. Function rcu\_usart\_clock\_config**

<b>Function name</b>	rcu_usart_clock_config
<b>Function prototype</b>	void rcu_usart_clock_config(usart_idx_enum usart_idx, uint32_t ck_usart);
<b>Function descriptions</b>	configure the USARTx(x=0,1,2) clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_idx</b>	USART index, refer to <a href="#">Table 3-840. Enum usart_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_usart</b>	USART clock source selection
<i>RCU_USARTSRC_APB</i>	select CK_APB as USART clock
<i>RCU_USARTSRC_CK_SYS</i>	select CK_SYS as USART clock
<i>RCU_USARTSRC_LXTAL</i>	select CK_LXTAL as USART clock

<i>RCU_USARTSRC_IRC8M</i>	select CK_IRC8M as USART clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the LXTAL as USART0 clock */
```

```
rcu_usart_clock_config(IDX_USART0, RCU_USARTSRC_LXTAL);
```

### rcu\_i2c\_clock\_config

The description of rcu\_i2c\_clock\_config is shown as below:

**Table 3-869. Function rcu\_i2c\_clock\_config**

<b>Function name</b>	rcu_i2c_clock_config
<b>Function prototype</b>	void rcu_i2c_clock_config(i2c_idx_enum i2c_idx, uint32_t ck_i2c);
<b>Function descriptions</b>	configure the I2Cx(x=0,1,2,3) clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_idx</b>	I2C index, refer to <a href="#">Table 3-841. Enum i2c_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_i2c</b>	I2C clock source selection
<i>RCU_I2CSRC_APB1</i>	select CK_APB1 as I2C clock
<i>RCU_I2CSRC_CKSYS</i>	select CK_SYS as I2C clock
<i>RCU_I2CSRC_IRC8M</i>	select CK_IRC8M as I2C clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK_APB1 as I2C0 clock */
```

```
rcu_i2c_clock_config(IDX_I2C0, RCU_I2CSRC_APB1);
```

### rcu\_can\_clock\_config

The description of rcu\_i2c\_clock\_config is shown as below:

**Table 3-870. Function rcu\_can\_clock\_config**

<b>Function name</b>	rcu_can_clock_config
----------------------	----------------------

<b>Function prototype</b>	void rcu_can_clock_config(can_idx_enum can_idx, uint32_t ck_can);
<b>Function descriptions</b>	configure the CANx(x=0,1,2) clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_idx</b>	CAN index, refer to <a href="#">Table 3-842. Enum can_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_can</b>	CAN clock source selection
<i>RCU_CANSRC_IRC8M</i>	select CK_IRC8M as CAN clock
<i>RCU_CANSRC_APB2</i>	select CK_APB2 as CAN clock
<i>RCU_CANSRC_PLLQ</i>	select PLLQ as CAN clock
<i>RCU_CANSRC_HXTAL</i> <i>L</i>	select HXTAL as CAN clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK_IRC8M as the CAN0 clock source */
```

```
rcu_can_clock_config(IDX_CAN0, RCU_CANSRC_IRC8M);
```

### rcu\_adc\_clock\_config

The description of rcu\_adc\_clock\_config is shown as below:

**Table 3-871. Function rcu\_adc\_clock\_config**

<b>Function name</b>	rcu_adc_clock_config
<b>Function prototype</b>	void rcu_adc_clock_config(adc_idx_enum adc_idx, uint32_t ck_adc);
<b>Function descriptions</b>	configure the ADCx(x=0,1,2,3) clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_idx</b>	ADC index, refer to <a href="#">Table 3-843. Enum adc_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_adc</b>	ADC clock source selection
<i>RCU_ADCSRC_PLLR</i>	select CK_PLLR as ADC clock
<i>RCU_ADCSRC_CKSYS</i> <i>S</i>	select CK_SYS as ADC clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK_PLLR as the ADC0 clock source */
rcu_adc_clock_config(IDX_ADC0, CK_PLLR);
```

### rcu\_hpdf\_clock\_config

The description of rcu\_hpdf\_clock\_config is shown as below:

**Table 3-872. Function rcu\_hpdf\_clock\_config**

<b>Function name</b>	rcu_hpdf_clock_config
<b>Function prototype</b>	void rcu_hpdf_clock_config(uint32_t ck_hpdf);
<b>Function descriptions</b>	configure the HPDF clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_hpdf</b>	HPDF clock source selection
<i>RCU_HPDFSRC_APB2</i>	select CK_APB2 as HPDF clock
<i>RCU_HPDFSRC_AHB</i>	select CK_AHB as HPDF clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK_AHB as the HPDF clock source */
rcu_hpdf_clock_config(CK_AHB);
```

### rcu\_hpdfaudio\_clock\_config

The description of rcu\_hpdfaudio\_clock\_config is shown as below:

**Table 3-873. Function rcu\_hpdfaudio\_clock\_config**

<b>Function name</b>	rcu_hpdfaudio_clock_config
<b>Function prototype</b>	void rcu_hpdfaudio_clock_config(uint32_t ck_hpdfaudio);
<b>Function descriptions</b>	configure the HPDF clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_hpdfaudio</b>	HPDF clock source selection
<i>RCU_HPDAUDIO_PL LQ</i>	CK_HPDAUDIO select CK_PLLQ
<i>RCU_HPDAUDIO_EH PDAUDIOPIN</i>	CK_HPDAUDIO select CK_EHPDAUDIOPIN



<i>RCU_HPDAUDIO_IR</i> <i>C8M</i>	CK_HPDAUDIO select IRC8M
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the PLLQ as the HPDAUDIO clock source */
rcu_hpdaudio_clock_config (RCU_HPDAUDIO_PLLQ);
```

### rcu\_trng\_clock\_config

The description of rcu\_trng\_clock\_config is shown as below:

**Table 3-874. Function rcu\_trng\_clock\_config**

<b>Function name</b>	rcu_trng_clock_config
<b>Function prototype</b>	void rcu_trng_clock_config(uint32_t ck_trng);
<b>Function descriptions</b>	configure the TRNG prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_trng</b>	TRNG clock prescaler selection
<i>RCU_TRNG_CKPLLQ_DIVx</i> ( <i>x</i> = 2, 3, 4,... , 15)	TRNG clock is CKPLLQ/ <i>x</i>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TRNG prescaler selection */
rcu_trng_clock_config (RCU_TRNG_CKPLLQ_DIV3);
```

### rcu\_lptimer\_clock\_config

The description of rcu\_lptimer\_clock\_config is shown as below:

**Table 3-875. Function rcu\_lptimer\_clock\_config**

<b>Function name</b>	rcu_lptimer_clock_config
<b>Function prototype</b>	void rcu_lptimer_clock_config(uint32_t ck_lptimer);
<b>Function descriptions</b>	configure the LPTIMER clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>ck_lptimer</b>	LPTIMER clock source selection
<i>RCU_LPTIMERSRC_CKAPB1</i>	CK_LPTIMER select CK_APB1
<i>RCU_LPTIMERSRC_IRC32K</i>	CK_LPTIMER select CK_IRC32K
<i>RCU_LPTIMERSRC_LXTAL</i>	CK_LPTIMER select CK_LXTAL
<i>RCU_LPTIMERSRC_IRC8M</i>	CK_LPTIMER select CK_IRC8M
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_APB1 as the LPTIMER clock source */
```

```
rcu_lptimer_clock_config(RCU_LPTIMERSRC_CKAPB1);
```

### rcu\_qspi\_clock\_config

The description of rcu\_qspi\_clock\_config is shown as below:

**Table 3-876. Function rcu\_qspi\_clock\_config**

<b>Function name</b>	rcu_qspi_clock_config
<b>Function prototype</b>	void rcu_qspi_clock_config (uint32_t ck_qspi);
<b>Function descriptions</b>	configure the QSPI clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>ck_qspi</b>	QSPI clock source selection
<i>RCU_QSPISRC_CKSYS</i>	CK_QSPI select CK_CKSYS
<i>RCU_QSPISRC_IRC8M</i>	CK_QSPI select CK_IRC8M
<i>RCU_QSPISRC_PLLQ</i>	CK_QSPI select CK_PLLQ
<i>RCU_QSPISRC_PLLR</i>	CK_QSPI select CK_PLLR
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_SYS as the QSPI clock source */
```

```
rcu_qspi_clock_config (RCU_QSPISRC_CKSYS);
```

### rcu\_hrtimer\_clock\_config

The description of rcu\_hrtimer\_clock\_config is shown as below:

**Table 3-877. Function rcu\_hrtimer\_clock\_config**

<b>Function name</b>	rcu_hrtimer_clock_config
<b>Function prototype</b>	void rcu_hrtimer_clock_config(uint32_t hrtimer_clock_source);
<b>Function descriptions</b>	configure the HRTIMER clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_clock_source</b>	HRTIMER clock source selection
<i>RCU_HRTIMERSRC_CKAPB2</i>	APB2 clock selected as HRTIMER source clock
<i>RCU_HRTIMERSRC_CKSYS</i>	system clock selected as HRTIMER source clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the APB2 as the HRTIMER clock source */
```

```
rcu_hrtimer_clock_config (RCU_HRTIMERSRC_CKAPB2);
```

### rcu\_lxtal\_drive\_capability\_config

The description of rcu\_lxtal\_drive\_capability\_config is shown as below:

**Table 3-878. Function rcu\_lxtal\_drive\_capability\_config**

<b>Function name</b>	rcu_lxtal_drive_capability_config
<b>Function prototype</b>	void rcu_lxtal_drive_capability_config(uint32_t lxtal_dricap);
<b>Function descriptions</b>	configure the LXTAL drive capability
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lxtal_dricap</b>	drive capability of LXTAL
<i>RCU_LXTAL_LOWDRI</i>	lower driving capability
<i>RCU_LXTAL_MED_LOWDRI</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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the LXTAL lower driving capability */
```

```
rcu_lxtal_drive_capability_config(RCU_LXTAL_LOWDRI);
```

### rcu\_osc\_i\_stab\_wait

The description of rcu\_osc\_i\_stab\_wait is shown as below:

**Table 3-879. Function rcu\_osc\_i\_stab\_wait**

<b>Function name</b>	rcu_osc_i_stab_wait
<b>Function prototype</b>	ErrStatus rcu_osc_i_stab_wait(rcu_osc_type_enum osci);
<b>Function descriptions</b>	wait for oscillator stabilization flags is SET or oscillator startup is timeout
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>osci</b>	oscillator types, refer to <a href="#">Table 3-838. Enum rcu_osc_type_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```
/* wait for oscillator stabilization flag */
```

```
if(SUCCESS == rcu_osc_i_stab_wait(RCU_HXTAL)){  
    }  
}
```

### rcu\_osc\_i\_on

The description of rcu\_osc\_i\_on is shown as below:

**Table 3-880. Function rcu\_osc\_i\_on**

<b>Function name</b>	rcu_osc_i_on
<b>Function prototype</b>	void rcu_osc_i_on(rcu_osc_type_enum osci);
<b>Function descriptions</b>	turn on the oscillator
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
osci	oscillator types, refer to <a href="#">Table 3-838. Enum rcu_osci_type_enum</a>
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 3-881. 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 <a href="#">Table 3-838. Enum rcu_osci_type_enum</a>
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 3-882. 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}	
<b>osci</b>	oscillator types, refer to <a href="#">Table 3-838. Enum rcu_osci_type_enum</a>
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 3-883. Function rcu\_osci\_bypass\_mode\_disable**

<b>Function name</b>	rcu_osci_bypass_mode_disable
<b>Function prototype</b>	void rcu_osci_bypass_mode_disable(rcu_osci_type_enum osci);
<b>Function descriptions</b>	disable the oscillator bypass mode
<b>Precondition</b>	HXTALEN or LXTALEN must be reset before it
<b>The called functions</b>	-
Input parameter{in}	
<b>osci</b>	oscillator types, refer to <a href="#">Table 3-838. Enum rcu_osci_type_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the high speed crystal oscillator bypass mode */
```

```
rcu_osci_bypass_mode_disable(RCU_HXTAL);
```

### rcu\_irc8m\_adjust\_value\_set

The description of rcu\_irc8m\_adjust\_value\_set is shown as below:

**Table 3-884. Function rcu\_irc8m\_adjust\_value\_set**

<b>Function name</b>	rcu_irc8m_adjust_value_set
<b>Function prototype</b>	void rcu_irc8m_adjust_value_set(uint32_t irc8m_adjval);
<b>Function descriptions</b>	set the IRC8M adjust value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	

<b>irc8m_adjval</b>	IRC8M adjust value, must be between 0 and 0x1F
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the IRC8M adjust value */
rcu_irc8m_adjust_value_set(0x20);
```

### rcu\_hxtal\_clock\_monitor\_enable

The description of rcu\_hxtal\_clock\_monitor\_enable is shown as below:

**Table 3-885. Function rcu\_hxtal\_clock\_monitor\_enable**

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

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 3-886. Function rcu\_hxtal\_clock\_monitor\_disable**

<b>Function name</b>	rcu_hxtal_clock_monitor_disable
<b>Function prototype</b>	void rcu_hxtal_clock_monitor_disable(void);
<b>Function descriptions</b>	disable the HXTAL clock monitor
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-

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 3-887. 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 3-888. 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\_clock\_freq\_get

The description of rcu\_clock\_freq\_get is shown as below:

**Table 3-889. Function rcu\_clock\_freq\_get**

<b>Function name</b>	rcu_clock_freq_get
<b>Function prototype</b>	uint32_t rcu_clock_freq_get(rcu_clock_freq_enum clock);
<b>Function descriptions</b>	get the system clock, bus clock and peripheral clock frequency
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
clock	the clock frequency which to get, refer to <a href="#">Table 3-839. Enum rcu_clock_freq_enum</a>
<b>Output parameter{out}</b>	
-	
<b>Return value</b>	
uint32_t	clock frequency of system, AHB, APB1, APB2, APB3, APB4, PLL 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 3-890. Function rcu\_flag\_get**

<b>Function name</b>	rcu_flag_get
<b>Function prototype</b>	FlagStatus rcu_flag_get(rcu_flag_enum flag);
<b>Function descriptions</b>	get the clock stabilization and peripheral reset flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

flag	the clock stabilization and peripheral reset flags, refer to <a href="#">Table 3-834. Enum rcu_flag_enum</a>
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 3-891. 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\_enable

The description of rcu\_interrupt\_enable is shown as below:

**Table 3-892. 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}	
<b>interrupt</b>	clock stabilization interrupt, refer to <a href="#">Table 3-837. Enum rcu_int_enum</a>
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 3-893. Function rcu\_interrupt\_disable**

<b>Function name</b>	rcu_interrupt_disable
<b>Function prototype</b>	void rcu_interrupt_disable(rcu_int_enum interrupt);
<b>Function descriptions</b>	disable the stabilization interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>interrupt</b>	clock stabilization interrupt, refer to <a href="#">Table 3-837. Enum rcu_int_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### rcu\_interrupt\_flag\_get

The description of rcu\_interrupt\_flag\_get is shown as below:

**Table 3-894. Function rcu\_interrupt\_flag\_get**

<b>Function name</b>	rcu_interrupt_flag_get
<b>Function prototype</b>	FlagStatus rcu_interrupt_flag_get(rcu_int_flag_enum int_flag);
<b>Function descriptions</b>	get the clock stabilization interrupt and ckm flags
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	

<b>int_flag</b>	interrupt and ckm flags, refer to <a href="#">Table 3-835. Enum rcu_int_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-895. Function rcu\_interrupt\_flag\_clear**

<b>Function name</b>	rcu_interrupt_flag_clear
<b>Function prototype</b>	void rcu_interrupt_flag_clear(rcu_int_flag_clear_enum int_flag)
<b>Function descriptions</b>	clear the interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	clock stabilization and stuck interrupt flags clear, refer to <a href="#">Table 3-836. Enum rcu_int_flag_clear_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## 3.26. 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 [3.26.1](#), the FWDGT firmware functions are introduced in chapter [3.26.2](#).

### 3.26.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

**Table 3-896. RTC Registers**

Registers	Descriptions
RTC_TIME	RTC time of day register
RTC_DATE	RTC date register
RTC_CTL	RTC control register
RTC_STAT	RTC status register
RTC_PSC	RTC time prescaler register
RTC_WUT	RTC wakeup timer register
RTC_ALRM0TD	RTC alarm 0 time and date register
RTC_ALRM1TD	RTC alarm 1 time and date register
RTC_WPK	RTC write protection key register
RTC_SS	RTC sub second register
RTC_SHIFTCTL	RTC shift function control register
RTC_TTS	RTC time of timestamp register
RTC_DTS	RTC date of timestamp register
RTC_SSTS	RTC sub second of timestamp register
RTC_HRFC	RTC high resolution frequency compensation register
RTC_TAMP	RTC tamper register
RTC_ALRM0SS	RTC alarm 0 sub second register
RTC_ALRM1SS	RTC alarm 1 sub second register
RTC_CFG	RTC configure register
RTC_BKP0	RTC backup 0 register
RTC_BKP1	RTC backup 1 register
RTC_BKP2	RTC backup 2 register
RTC_BKP3	RTC backup 3 register
RTC_BKP4	RTC backup 4 register
RTC_BKP5	RTC backup 5 register
RTC_BKP6	RTC backup 6 register
RTC_BKP7	RTC backup 7 register
RTC_BKP8	RTC backup 8 register
RTC_BKP9	RTC backup 9 register
RTC_BKP10	RTC backup 10 register
RTC_BKP11	RTC backup 11 register
RTC_BKP12	RTC backup 12 register
RTC_BKP13	RTC backup 13 register
RTC_BKP14	RTC backup 14 register
RTC_BKP15	RTC backup 15 register
RTC_BKP16	RTC backup 16 register
RTC_BKP17	RTC backup 17 register

Registers	Descriptions
RTC_BKP18	RTC backup 18 register
RTC_BKP19	RTC backup 19 register
RTC_BKP20	RTC backup 20 register
RTC_BKP21	RTC backup 21 register
RTC_BKP22	RTC backup 22 register
RTC_BKP23	RTC backup 23register
RTC_BKP24	RTC backup 24 register
RTC_BKP25	RTC backup 25 register
RTC_BKP26	RTC backup 26 register
RTC_BKP27	RTC backup 27 register
RTC_BKP28	RTC backup 28 register
RTC_BKP29	RTC backup 29 register
RTC_BKP30	RTC backup 30 register
RTC_BKP31	RTC backup 31 register

### 3.26.2. Descriptions of Peripheral functions

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

**Table 3-897. RTC firmware function**

Function name	Function description
rtc_deinit	reset most of the RTC registers
rtc_init	initialize RTC registers
rtc_init_mode_enter	enter RTC init mode
rtc_init_mode_exit	exit RTC init mode
rtc_register_sync_wait	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
rtc_current_time_get	get current time and date
rtc_subsecond_get	get current subsecond value
rtc_alarm_config	configure RTC alarm
rtc_alarm_subsecond_config	configure subsecond of RTC alarm
rtc_alarm_get	get RTC alarm
rtc_alarm_subsecond_get	get RTC alarm subsecond
rtc_alarm_enable	enable RTC alarm
rtc_alarm_disable	disable RTC alarm
rtc_timestamp_enable	enable RTC time-stamp
rtc_timestamp_disable	disable RTC time-stamp
rtc_timestamp_get	get RTC timestamp time and date
rtc_timestamp_internalevent_config	configure RTC time-stamp internal event
rtc_timestamp_subsecond_get	get RTC time-stamp subsecond
rtc_tamper_enable	enable RTC tamper

Function name	Function description
rtc_tamper_disable	disable RTC tamper
rtc_output_pin_select	select the RTC output pin
rtc_alarm_output_config	configure RTC alarm output source
rtc_calibration_output_config	configure RTC calibration output source
rtc_hour_adjust	adjust the daylight saving time by adding or subtracting one hour from the current time
rtc_second_adjust	adjust RTC second or subsecond value of current time
rtc_bypass_shadow_enable	enable RTC bypass shadow registers function
rtc_bypass_shadow_disable	disable RTC bypass shadow registers function
rtc_refclock_detection_enable	enable RTC reference clock detection function
rtc_refclock_detection_disable	disable RTC reference clock detection function
rtc_wakeup_enable	enable RTC wakeup timer
rtc_wakeup_disable	disable RTC wakeup timer
rtc_wakeup_clock_set	set auto wakeup timer clock
rtc_wakeup_timer_set	set auto wakeup timer value
rtc_wakeup_timer_get	get auto wakeup timer value
rtc_smooth_calibration_config	configure RTC smooth calibration
rtc_interrupt_enable	enable specified RTC interrupt
rtc_interrupt_disable	disable specified RTC interrupt
rtc_flag_get	check specified flag
rtc_flag_clear	clear specified flag

### Structure rtc\_parameter\_struct

**Table 3-898. Structure rtc\_parameter\_struct**

Member name	Function description
year	RTC year value: 0x0 - 0x99(BCD format)
month	RTC month value (BCD format)
date	RTC date value: 0x1 - 0x31(BCD format)
day_of_week	RTC weekday value(BCD format)
hour	RTC hour value: 0x1 - 0x12(BCD format) or 0x0 - 0x23(BCD format)
minute	RTC minute value: 0x0 - 0x59(BCD format)
second	RTC second value: 0x0 - 0x59(BCD format)
factor_asyn	RTC asynchronous prescaler value: 0x0 - 0x7F
factor_syn	RTC synchronous prescaler value: 0x0 - 0x7FFF
am_pm	RTC AM/PM value
display_format	RTC time notation

### Structure rtc\_alarm\_struct

**Table 3-899. Structure rtc\_alarm\_struct**

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

alarm_mask	RTC alarm mask
weekday_or_date	specify RTC alarm is on date or weekday
alarm_day	RTC alarm date or weekday value(BCD format)
alarm_hour	RTC alarm hour value: 0x1 - 0x12(BCD format) or 0x0 - 0x23(BCD format)
alarm_minute	RTC alarm minute value: 0x0 - 0x59(BCD format)
alarm_second	RTC alarm second value: 0x0 - 0x59(BCD format)
am_pm	RTC alarm AM/PM value

### Structure rtc\_timestamp\_struct

**Table 3-900. Structure rtc\_timestamp\_struct**

Member name	Function description
timestamp_month	RTC time-stamp month value(BCD format)
timestamp_date	RTC time-stamp date value: 0x1 - 0x31(BCD format)
timestamp_day	RTC time-stamp weekday value(BCD format)
timestamp_hour	RTC time-stamp hour value(BCD format): 0x1 - 0x12(BCD format) or 0x0 - 0x23(BCD format)
timestamp_minute	RTC time-stamp minute value: 0x0 - 0x59(BCD format)
timestamp_second	RTC time-stamp second value: 0x0 - 0x59(BCD format)
am_pm	RTC time-stamp AM/PM value

### Structure rtc\_tamper\_struct

**Table 3-901. Structure rtc\_tamper\_struct**

Member name	Function description
tamper_source	RTC tamper source
tamper_trigger	RTC tamper trigger
tamper_filter	RTC tamper consecutive samples needed during a voltage level detection
tamper_sample_frequency	RTC tamper sampling frequency during a voltage level detection
tamper_precharge_enable	RTC tamper precharge feature during a voltage level detection
tamper_precharge_time	RTC tamper precharge duration if precharge feature is enabled
tamper_with_timestamp	RTC tamper time-stamp feature

### rtc\_deinit

The description of rtc\_deinit is shown as below:

**Table 3-902. Function rtc\_deinit**

Function name	rtc_deinit
Function prototype	ErrStatus rtc_deinit(void);



<b>Function descriptions</b>	reset most of the RTC registers
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable/ rcu_periph_reset_disable -
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* reset most of the RTC registers*/
```

```
ErrStatus error_status = rtc_deinit();
```

## rtc\_init

The description of rtc\_init is shown as below:

**Table 3-903. Function rtc\_init**

<b>Function name</b>	rtc_init
<b>Function prototype</b>	ErrStatus rtc_init(rtc_parameter_struct* rtc_initpara_struct);
<b>Function descriptions</b>	initialize RTC registers
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_initpara_struct</b>	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure <a href="#">Table 3-898. Structure rtc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* initialize RTC registers */
```

```
rtc_parameter_struct rtc_initpara;
```

```
rtc_interrupt_disable(RTC_INT_SECOND);
```

```
rtc_initpara.factor_asyn = prescaler_a;
```

```
rtc_initpara.factor_syn = prescaler_s;
```

```
rtc_initpara.year = 0x16;
```

```

rtc_initpara.day_of_week = RTC_SATURDAY;

rtc_initpara.month = RTC_APR;

rtc_initpara.date = 0x30;

rtc_initpara.display_format = RTC_24HOUR;

rtc_initpara.am_pm = RTC_AM;

rtc_init(&rtc_initpara);

```

### rtc\_init\_mode\_enter

The description of rtc\_init\_mode\_enter is shown as below:

**Table 3-904. Function rtc\_init\_mode\_enter**

<b>Function name</b>	rtc_init_mode_enter
<b>Function prototype</b>	ErrStatus rtc_init_mode_enter(void);
<b>Function descriptions</b>	enter RTC init mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```

/*enter RTC init mode*/

ErrStatus error_status = rtc_init_mode_enter();

```

### rtc\_init\_mode\_exit

The description of rtc\_init\_mode\_exit is shown as below:

**Table 3-905. Function rtc\_init\_mode\_exit**

<b>Function name</b>	rtc_init_mode_exit
<b>Function prototype</b>	void rtc_init_mode_exit(void);
<b>Function descriptions</b>	exit RTC init mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/*exit RTC init mode*/
```

```
rtc_init_mode_exit();
```

### rtc\_register\_sync\_wait

The description of rtc\_register\_sync\_wait is shown as below:

**Table 3-906. Function rtc\_register\_sync\_wait**

<b>Function name</b>	rtc_register_sync_wait
<b>Function prototype</b>	ErrStatus rtc_register_sync_wait(void);
<b>Function descriptions</b>	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/*wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated*/
```

```
ErrStatus error_status = rtc_register_sync_wait();
```

### rtc\_current\_time\_get

The description of rtc\_current\_time\_get is shown as below:

**Table 3-907. Function rtc\_current\_time\_get**

<b>Function name</b>	rtc_current_time_get
<b>Function prototype</b>	void rtc_current_time_get(rtc_parameter_struct* rtc_initpara_struct);
<b>Function descriptions</b>	get current time and date
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	

<b>rtc_initpara_struct</b>	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure <a href="#">Table 3-898. Structure rtc_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

```
/*get current time and date*/

rtc_parameter_struct rtc_initpara_struct;

rtc_current_time_get(&rtc_initpara_struct);
```

### rtc\_subsecond\_get

The description of rtc\_subsecond\_get is shown as below:

**Table 3-908. Function rtc\_subsecond\_get**

<b>Function name</b>	rtc_subsecond_get
<b>Function prototype</b>	uint32_t rtc_subsecond_get(void);
<b>Function descriptions</b>	get current subsecond value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	current subsecond value(0x00-0xFFFF)

Example:

```
/*get current subsecond value*/

uint32_t sub_second = rtc_subsecond_get();
```

### rtc\_alarm\_config

The description of rtc\_alarm\_config is shown as below:

**Table 3-909. Function rtc\_alarm\_config**

<b>Function name</b>	rtc_alarm_config
<b>Function prototype</b>	void rtc_alarm_config(uint8_t rtc_alarm, rtc_alarm_struct *rtc_alarm_time);
<b>Function descriptions</b>	configure RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Input parameter{in}</b>	
<b>rtc_alarm_time</b>	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure <a href="#">Table 3-899. Structure rtc_alarm_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*rtc_alarm_config*/
```

```
rtc_alarm_struct rtc_alarm_time;
```

```
rtc_alarm_config(RTC_ALARM0,&rtc_alarm_time);
```

### rtc\_alarm\_subsecond\_config

The description of rtc\_alarm\_subsecond\_config is shown as below:

**Table 3-910. Function rtc\_alarm\_subsecond\_config**

<b>Function name</b>	rtc_alarm_subsecond_config
<b>Function prototype</b>	void rtc_alarm_subsecond_config(uint8_t rtc_alarm, uint32_t mask_subsecond, uint32_t subsecond)
<b>Function descriptions</b>	configure subsecond of RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Input parameter{in}</b>	
<b>mask_subsecond</b>	alarm subsecond mask
<i>RTC_MASKSSC_0_14</i>	mask alarm subsecond configuration
<i>RTC_MASKSSC_1_14</i>	mask RTC_ALRM0SS_SSC[14:1], and RTC_ALRM0SS_SSC[0] is to be compared
<i>RTC_MASKSSC_2_14</i>	mask RTC_ALRM0SS_SSC[14:2], and RTC_ALRM0SS_SSC[1:0] is to be compared
<i>RTC_MASKSSC_3_14</i>	mask RTC_ALRM0SS_SSC[14:3], and RTC_ALRM0SS_SSC[2:0] is to be compared
<i>RTC_MASKSSC_4_14</i>	mask RTC_ALRM0SS_SSC[14:4], and RTC_ALRM0SS_SSC[3:0] is to be compared
<i>RTC_MASKSSC_5_14</i>	mask RTC_ALRM0SS_SSC[14:5], and RTC_ALRM0SS_SSC[4:0] is to be compared
<i>RTC_MASKSSC_6_14</i>	mask RTC_ALRM0SS_SSC[14:6], and RTC_ALRM0SS_SSC[5:0] is to be compared

<i>RTC_MASKSSC_7_14</i>	mask RTC_ALARM0SS_SSC[14:7], and RTC_ALARM0SS_SSC[6:0] is to be compared
<i>RTC_MASKSSC_8_14</i>	mask RTC_ALARM0SS_SSC[14:8], and RTC_ALARM0SS_SSC[7:0] is to be compared
<i>RTC_MASKSSC_9_14</i>	mask RTC_ALARM0SS_SSC[14:9], and RTC_ALARM0SS_SSC[8:0] is to be compared
<i>RTC_MASKSSC_10_14</i>	mask RTC_ALARM0SS_SSC[14:10], and RTC_ALARM0SS_SSC[9:0] is to be compared
<i>RTC_MASKSSC_11_14</i>	mask RTC_ALARM0SS_SSC[14:11], and RTC_ALARM0SS_SSC[10:0] is to be compared
<i>RTC_MASKSSC_12_14</i>	mask RTC_ALARM0SS_SSC[14:12], and RTC_ALARM0SS_SSC[11:0] is to be compared
<i>RTC_MASKSSC_13_14</i>	mask RTC_ALARM0SS_SSC[14:13], and RTC_ALARM0SS_SSC[12:0] is to be compared
<i>RTC_MASKSSC_14</i>	mask RTC_ALARM0SS_SSC[14], and RTC_ALARM0SS_SSC[13:0] is to be compared
<i>RTC_MASKSSC_NONE</i>	mask none, and RTC_ALARM0SS_SSC[14:0] is to be compared
<b>Input parameter{in}</b>	
<b>subsecond</b>	alarm subsecond value(0x000 - 0x7FFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*configure subsecond of RTC alarm*/
```

```
rtc_subsecond_config(RTC_ALARM0,RTC_MASKSSC_9_14, 0x7FFF);
```

### rtc\_alarm\_enable

The description of rtc\_alarm\_enable is shown as below:

**Table 3-911. Function rtc\_alarm\_enable**

<b>Function name</b>	rtc_alarm_enable
<b>Function prototype</b>	void rtc_alarm_enable(uint8_t rtc_alarm);
<b>Function descriptions</b>	enable RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/*enable RTC alarm*/
```

```
rtc_alarm_enable(RTC_ALARM0);
```

### rtc\_alarm\_disable

The description of rtc\_alarm\_disable is shown as below:

**Table 3-912. Function rtc\_alarm\_disable**

<b>Function name</b>	rtc_alarm_disable
<b>Function prototype</b>	ErrStatus rtc_alarm_disable(uint8_t rtc_alarm);
<b>Function descriptions</b>	disable RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/*disable RTC alarm*/
```

```
ErrStatus error_status = rtc_alarm_disable(RTC_ALARM0);
```

### rtc\_alarm\_get

The description of rtc\_alarm\_get is shown as below:

**Table 3-913. Function rtc\_alarm\_get**

<b>Function name</b>	rtc_alarm_get
<b>Function prototype</b>	void rtc_alarm_get(uint8_t rtc_alarm, rtc_alarm_struct *rtc_alarm_time);
<b>Function descriptions</b>	get RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Output parameter{out}</b>	
<b>rtc_alarm_time</b>	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to

	members of the structure <a href="#">Table 3-899. Structure rtc_alarm_struct</a>
<b>Return value</b>	
-	-

Example:

```
/* get RTC alarm */

rtc_alarm_struct rtc_alarm_time;

rtc_alarm_get(RTC_ALARM0,&rtc_alarm_time);
```

### rtc\_alarm\_subsecond\_get

The description of rtc\_alarm\_subsecond\_get is shown as below:

**Table 3-914. Function rtc\_alarm\_subsecond\_get**

<b>Function name</b>	rtc_alarm_subsecond_get
<b>Function prototype</b>	uint32_t rtc_alarm_subsecond_get(uint8_t rtc_alarm);
<b>Function descriptions</b>	get RTC alarm subsecond
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	RTC alarm subsecond value(0x0-0x3FFF)

Example:

```
/*get RTC alarm subsecond*/

uint32_t subsecond = rtc_alarm_subsecond_get(RTC_ALARM0);
```

### rtc\_timestamp\_enable

The description of rtc\_timestamp\_enable is shown as below:

**Table 3-915. Function rtc\_timestamp\_enable**

<b>Function name</b>	rtc_timestamp_enable
<b>Function prototype</b>	void rtc_timestamp_enable(uint32_t edge);
<b>Function descriptions</b>	enable RTC time-stamp
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>edge</b>	specify which edge to detect of time-stamp
<b>RTC_TIMESTAMP_RIS</b>	rising edge is valid event edge for timestamp event



ING_EDGE	
RTC_TIMESTAMP_FALLING_EDGE	falling edge is valid event edge for timestamp event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*enable RTC time-stamp*/
rtc_timestamp_enable(RTC_TIMESTAMP_RISING_EDGE);
```

### rtc\_timestamp\_disable

The description of rtc\_timestamp\_disable is shown as below:

**Table 3-916. Function rtc\_timestamp\_disable**

Function name	rtc_timestamp_disable
Function prototype	void rtc_timestamp_disable(void);
Function descriptions	disable RTC time-stamp
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*disable RTC time-stamp*/
rtc_timestamp_disable();
```

### rtc\_timestamp\_internalevent\_config

The description of rtc\_timestamp\_internalevent\_config is shown as below:

**Table 3-917. Function rtc\_timestamp\_internalevent\_config**

Function name	rtc_timestamp_internalevent_config
Function prototype	void rtc_timestamp_internalevent_config(uint32_t mode)
Function descriptions	configure RTC time-stamp internal event
Precondition	-
The called functions	-

Input parameter{in}	
<b>mode</b>	specify which internal or external event to be detected
<i>RTC_ITSEN_DISABLE</i>	disable RTC time-stamp internal event
<i>RTC_ITSEN_ENABLE</i>	enable RTC time-stamp internal event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RTC time-stamp internal event */
```

```
rtc_timestamp_internalevent_config(RTC_ITSEN_DISABLE);
```

### rtc\_timestamp\_get

The description of `rtc_timestamp_get` is shown as below:

**Table 3-918. Function `rtc_timestamp_get`**

<b>Function name</b>	<code>rtc_timestamp_get</code>
<b>Function prototype</b>	<code>void rtc_timestamp_get(rtc_timestamp_struct* rtc_timestamp);</code>
<b>Function descriptions</b>	get RTC timestamp time and date
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
<code>rtc_timestamp</code>	Pointer to a <code>rtc_timestamp_struct</code> structure which contains parameters for RTC time-stamp configuration, the structure members can refer to members of the structure <a href="#">Table 3-901. Structure <code>rtc_tamper_struct</code></a>
Return value	
-	-

Example:

```
/* get RTC timestamp time and date */
```

```
rtc_timestamp_struct rtc_timestamp;
```

```
rtc_timestamp_get(& rtc_timestamp);
```

### rtc\_timestamp\_subsecond\_get

The description of `rtc_timestamp_subsecond_get` is shown as below:

**Table 3-919. Function `rtc_timestamp_subsecond_get`**

<b>Function name</b>	<code>rtc_timestamp_subsecond_get</code>
----------------------	--

<b>Function prototype</b>	uint32_t rtc_timestamp_subsecond_get(void);
<b>Function descriptions</b>	get RTC time-stamp subsecond
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	RTC time-stamp subsecond value

Example:

```
/* get RTC time-stamp subsecond */
uint32_t subsecond = rtc_timestamp_subsecond_get();
```

### rtc\_tamper\_enable

The description of rtc\_tamper\_enable is shown as below:

**Table 3-920. Function rtc\_tamper\_enable**

<b>Function name</b>	rtc_tamper_enable
<b>Function prototype</b>	void rtc_tamper_enable(rtc_tamper_struct* rtc_tamper);
<b>Function descriptions</b>	enable RTC tamper
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
rtc_tamper	pointer to a rtc_tamper_struct structure which contains parameters for RTC tamper configuration, the structure members can refer to members of the structure <a href="#">Table 3-901. Structure rtc_tamper_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable RTC tamper */
rtc_tamper_struct rtc_tamper
rtc_tamper_enable(& rtc_tamper);
```

### rtc\_tamper\_disable

The description of rtc\_tamper\_disable is shown as below:

Table 3-921. Function rtc\_tamper\_disable

<b>Function name</b>	rtc_tamper_disable
<b>Function prototype</b>	void rtc_tamper_disable(uint32_t source);
<b>Function descriptions</b>	disable RTC tamper
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>source</b>	specify which tamper source to be disabled
<i>RTC_TAMPER0</i>	RTC tamper0
<i>RTC_TAMPER1</i>	RTC tamper1
<i>RTC_TAMPER2</i>	RTC tamper2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable RTC tamper0 */
rtc_tamper_disable(RTC_TAMPER0);
```

### rtc\_output\_pin\_select

The description of rtc\_output\_pin\_select is shown as below:

Table 3-922. Function rtc\_output\_pin\_select

<b>Function name</b>	rtc_output_pin_select
<b>Function prototype</b>	void rtc_output_pin_select(uint32_t outputpin);
<b>Function descriptions</b>	select the RTC output pin
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>outputpin</b>	specify the rtc output pin is PC13 or PB2
<i>RTC_OUT_PC13</i>	the rtc output pin is PC13
<i>RTC_OUT_PB2</i>	the rtc output pin is PB2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* specify the rtc output pin is PC13 */
rtc_output_pin_select(RTC_OUT_PC13);
```

## rtc\_alarm\_output\_config

The description of rtc\_alarm\_output\_config is shown as below:

**Table 3-923. Function rtc\_alarm\_output\_config**

<b>Function name</b>	rtc_alarm_output_config
<b>Function prototype</b>	void rtc_alarm_output_config(uint32_t source, uint32_t mode);
<b>Function descriptions</b>	configure rtc alarm output source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>source</b>	specify signal to output
<i>RTC_ALARM0_HIGH</i>	when the alarm0 flag is set, the output pin is high
<i>RTC_ALARM0_LOW</i>	when the alarm0 flag is set, the output pin is low
<i>RTC_ALARM1_HIGH</i>	when the alarm1 flag is set, the output pin is high
<i>RTC_ALARM1_LOW</i>	when the alarm1 flag is set, the output pin is low
<i>RTC_WAKEUP_HIGH</i>	when the wakeup flag is set, the output pin is high
<i>RTC_WAKEUP_LOW</i>	when the wakeup flag is set, the output pin is low
<b>Input parameter{in}</b>	
<b>mode</b>	specify the output pin mode when output alarm signal or auto wakeup signal
<i>RTC_ALARM_OUTPUTPU</i> <i>T_OD</i>	open drain mode
<i>RTC_ALARM_OUTPUTPU</i> <i>T_PP</i>	push pull mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure rtc alarm0 output source */
rtc_alarm_output_config(RTC_ALARM0_LOW, RTC_ALARM_OUTPUT_PP);
```

## rtc\_calibration\_output\_config

The description of rtc\_calibration\_output\_config is shown as below:

**Table 3-924. Function rtc\_calibration\_output\_config**

<b>Function name</b>	rtc_calibration_output_config
<b>Function prototype</b>	void rtc_calibration_output_config(uint32_t source);
<b>Function descriptions</b>	configure rtc calibration output source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

source	specify signal to output
<i>RTC_CALIBRATION_5</i> <i>12HZ</i>	when the LSE frequency is 32768Hz and the RTC_PSC is the default value, output 512Hz signal
<i>RTC_CALIBRATION_1</i> <i>HZ</i>	when the LSE frequency is 32768Hz and the RTC_PSC is the default value, output 1Hz signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* when the LSE frequency is 32768Hz and the RTC_PSC
is the default value, output 1Hz signal */
rtc_calibration_output_config(RTC_CALIBRATION_1HZ);
```

### rtc\_hour\_adjust

The description of rtc\_hour\_adjust is shown as below:

**Table 3-925. Function rtc\_hour\_adjust**

Function name	rtc_hour_adjust
Function prototype	void rtc_hour_adjust(uint32_t operation);
Function descriptions	adjust the daylight saving time by adding or subtracting one hour from the current time
Precondition	-
The called functions	-
<b>Input parameter{in}</b>	
operation	hour ajustment operation
<i>RTC_CTL_A1H</i>	add one hour
<i>RTC_CTL_S1H</i>	subtract one hour
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* adjust the daylight saving time by adding one hour from the current time */
rtc_hour_adjust(RTC_CTL_A1H);
```

### rtc\_second\_adjust

The description of rtc\_second\_adjust is shown as below:

Table 3-926. Function rtc\_second\_adjust

Function name	rtc_second_adjust
Function prototype	ErrStatus rtc_second_adjust(uint32_t add, uint32_t minus);
Function descriptions	adjust RTC second or subsecond value of current time
Precondition	-
The called functions	-
Input parameter{in}	
add	add 1s to current time or not
RTC_SHIFT_ADD1S_RESET	no effect
RTC_SHIFT_ADD1S_SET	add 1s to current time
Input parameter{in}	
minus	number of subsecond to minus from current time(0x0 - 0x7FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* adjust RTC second or subsecond value of current time */
```

```
ErrStatus error_status = rtc_second_adjust(RTC_SHIFT_ADD1S_SET, 0);
```

### rtc\_bypass\_shadow\_enable

The description of rtc\_bypass\_shadow\_enable is shown as below:

Table 3-927. Function rtc\_bypass\_shadow\_enable

Function name	rtc_bypass_shadow_enable
Function prototype	void rtc_bypass_shadow_enable(void);
Function descriptions	enable RTC bypass shadow registers function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RTC bypass shadow registers function*/
```

```
rtc_bypass_shadow_enable();
```

## rtc\_bypass\_shadow\_disable

The description of rtc\_bypass\_shadow\_disable shown as below:

**Table 3-928. Function rtc\_bypass\_shadow\_disable**

<b>Function name</b>	rtc_bypass_shadow_disable
<b>Function prototype</b>	void rtc_bypass_shadow_disable(void);
<b>Function descriptions</b>	disable RTC bypass shadow registers function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable RTC bypass shadow registers function */
rtc_bypass_shadow_disable();
```

## rtc\_refclock\_detection\_enable

The description of rtc\_refclock\_detection\_enable shown as below:

**Table 3-929. Function rtc\_refclock\_detection\_enable**

<b>Function name</b>	rtc_refclock_detection_enable
<b>Function prototype</b>	ErrStatus rtc_refclock_detection_enable(void);
<b>Function descriptions</b>	enable RTC reference clock detection function
<b>Precondition</b>	-
<b>The called functions</b>	rtc_init_mode_enter/rtc_init_mode_exit
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* enable RTC reference clock detection function*/
ErrStatus error_status = rtc_refclock_detection_enable();
```



## rtc\_refclock\_detection\_disable

The description of rtc\_refclock\_detection\_disable is shown as below:

**Table 3-930. Function rtc\_refclock\_detection\_disable**

<b>Function name</b>	rtc_refclock_detection_disable
<b>Function prototype</b>	ErrStatus rtc_refclock_detection_disable(void);
<b>Function descriptions</b>	disable RTC reference clock detection function
<b>Precondition</b>	-
<b>The called functions</b>	rtc_init_mode_enter/rtc_init_mode_exit
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* disable RTC reference clock detection function */
ErrStatus error_status = rtc_refclock_detection_disable();
```

## rtc\_wakeup\_enable

The description of rtc\_refclock\_detection\_disable is shown as below:

**Table 3-931. Function rtc\_wakeup\_enable**

<b>Function name</b>	rtc_wakeup_enable
<b>Function prototype</b>	void rtc_wakeup_enable(void);
<b>Function descriptions</b>	enable RTC auto wakeup function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable RTC auto wakeup function */
rtc_wakeup_enable();
```

## rtc\_wakeup\_disable

The description of rtc\_wakeup\_disable is shown as below:

**Table 3-932. Function rtc\_wakeup\_disable**

<b>Function name</b>	rtc_wakeup_disable
<b>Function prototype</b>	ErrStatus rtc_wakeup_disable(void);
<b>Function descriptions</b>	disable RTC auto wakeup function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* disable RTC auto wakeup function */
ErrStatus error_status = rtc_wakeup_disable();
```

## rtc\_wakeup\_clock\_set

The description of rtc\_wakeup\_clock\_set is shown as below:

**Table 3-933. Function rtc\_wakeup\_clock\_set**

<b>Function name</b>	rtc_wakeup_clock_set
<b>Function prototype</b>	ErrStatus rtc_wakeup_clock_set(uint8_t wakeup_clock);
<b>Function descriptions</b>	set RTC auto wakeup timer clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wakeup_clock</b>	wakeup timer clock is RTC clock divided factor
WAKEUP_RTCK_DIV 16	RTC auto wakeup timer clock is RTC clock divided by 16
WAKEUP_RTCK_DIV 8	RTC auto wakeup timer clock is RTC clock divided by 8
WAKEUP_RTCK_DIV 4	RTC auto wakeup timer clock is RTC clock divided by 4
WAKEUP_RTCK_DIV 2	RTC auto wakeup timer clock is RTC clock divided by 2
WAKEUP_CKSPRE	RTC auto wakeup timer clock is ckspre
WAKEUP_CKSPRE_2 EXP16	RTC auto wakeup timer clock is ckspre and wakeup timer add 2exp16

Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* RTC auto wakeup timer clock is ckspre */
```

```
ErrStatus error_status = rtc_wakeup_clock_set(WAKEUP_RTCK_DIV8);
```

### rtc\_wakeup\_timer\_set

The description of rtc\_wakeup\_timer\_set is shown as below:

**Table 3-934. Function rtc\_wakeup\_timer\_set**

Function name	rtc_wakeup_timer_set
Function prototype	ErrStatus rtc_wakeup_timer_set(uint16_t wakeup_timer);
Function descriptions	set wakeup timer value
Precondition	-
The called functions	-
Input parameter{in}	
wakeup_timer	wakeup timer value
uint16_t	0x0000-0xffff
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* set wakeup timer value */
```

```
ErrStatus error_status = rtc_wakeup_timer_set(0XFFEE);
```

### rtc\_wakeup\_timer\_get

The description of rtc\_wakeup\_timer\_set is shown as below:

**Table 3-935. Function rtc\_wakeup\_timer\_get**

Function name	rtc_wakeup_timer_get
Function prototype	uint16_t rtc_wakeup_timer_get(void);
Function descriptions	set wakeup timer value
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
uint16_t	0-0xFFFF

Example:

```
/* get wakeup timer value */
uint32_t wakeup_time = rtc_wakeup_timer_get();
```

### rtc\_smooth\_calibration\_config

The description of rtc\_smooth\_calibration\_config is shown as below:

**Table 3-936. Function rtc\_smooth\_calibration\_config**

Function name	rtc_smooth_calibration_config
Function prototype	ErrStatus rtc_smooth_calibration_config(uint32_t window, uint32_t plus, uint32_t minus);
Function descriptions	configure RTC smooth calibration
Precondition	-
The called functions	-
Input parameter{in}	
window	select calibration window
RTC_CALIBRATION_WINDOW_32S	2exp20 RTCCLK cycles, 32s if RTCCLK = 32768 Hz
RTC_CALIBRATION_WINDOW_16S	2exp19 RTCCLK cycles, 16s if RTCCLK = 32768 Hz
RTC_CALIBRATION_WINDOW_8S	2exp18 RTCCLK cycles, 8s if RTCCLK = 32768 Hz
Input parameter{in}	
plus	add RTC clock or not
RTC_CALIBRATION_PLUS_SET	add one RTC clock every 2048 rtc clock
RTC_CALIBRATION_PLUS_RESET	no effect
Input parameter{in}	
minus	the RTC clock to minus during the calibration window(0x0 - 0xFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
/* configure RTC smooth calibration */
```

```
ErrStatus error_status;
```

```
error_status = rtc_smooth_calibration_config(RTC_CALIBRATION_WINDOW_32S,
RTC_CALIBRATION_PLUS_SET, 0x10);
```

### rtc\_interrupt\_enable

The description of rtc\_interrupt\_enable is shown as below:

**Table 3-937. Function rtc\_interrupt\_enable**

<b>Function name</b>	rtc_interrupt_enable
<b>Function prototype</b>	void rtc_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable specified RTC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	specify which interrupt source to be enabled
<i>RTC_INT_TIMESTAMP</i>	timestamp interrupt
<i>RTC_INT_ALARM0</i>	Alarm0 interrupt
<i>RTC_INT_ALARM1</i>	Alarm1 interrupt
<i>RTC_INT_TAMP_ALL</i>	tamp interrupt
<i>RTC_INT_TAMP0</i>	Tamper0 detection interrupt
<i>RTC_INT_TAMP1</i>	Tamper1 detection interrupt
<i>RTC_INT_TAMP2</i>	Tamper2 detection interrupt
<i>RTC_INT_WAKEUP</i>	wakeup timer interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable specified RTC interrupt*/
```

```
rtc_interrupt_enable(RTC_INT_TAMP0);
```

### rtc\_interrupt\_disable

The description of rtc\_interrupt\_disable is shown as below:

**Table 3-938. Function rtc\_interrupt\_disable**

<b>Function name</b>	rtc_interrupt_disable
<b>Function prototype</b>	void rtc_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disble specified RTC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>interrupt</b>	specify which RTC interrupt to disable
<i>RTC_INT_TIMESTAMP</i>	timestamp interrupt
<i>RTC_INT_ALARM0</i>	Alarm0 interrupt
<i>RTC_INT_ALARM1</i>	Alarm1 interrupt
<i>RTC_INT_TAMP_ALL</i>	All tamp interrupt
<i>RTC_INT_TAMP0</i>	Tamper0 detection interrupt
<i>RTC_INT_TAMP1</i>	Tamper1 detection interrupt
<i>RTC_INT_TAMP2</i>	Tamper2 detection interrupt
<i>RTC_INT_WAKEUP</i>	wakeup timer interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable specified RTC interrupt */
```

```
rtc_interrupt_disable(RTC_INT_TAMP0);
```

### rtc\_flag\_get

The description of rtc\_flag\_get is shown as below:

**Table 3-939. Function rtc\_flag\_get**

<b>Function name</b>	rtc_flag_get
<b>Function prototype</b>	FlagStatus rtc_flag_get(uint32_t flag);
<b>Function descriptions</b>	check specified flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>flag</b>	specify which flag to check
<i>RTC_FLAG_ALARM0</i> <i>W</i>	Alarm0 written available flag
<i>RTC_FLAG_ALARM1</i> <i>W</i>	Alarm1 written available flag
<i>RTC_FLAG_WTW</i>	wakeup timer can be written flag
<i>RTC_FLAG_SOP</i>	shift function operation pending flag
<i>RTC_FLAG_YCM</i>	year parameter configured event flag
<i>RTC_FLAG_RSYN</i>	time and date registers synchronized event flag
<i>RTC_FLAG_INIT</i>	init mode event flag
<i>RTC_FLAG_SCP</i>	smooth calibration pending flag
<i>RTC_FLAG_ALARM0</i>	Alarm0 event flag
<i>RTC_FLAG_ALARM1</i>	Alarm1 event flag

<i>RTC_FLAG_WT</i>	wakeup timer occurs flag
<i>RTC_FLAG_TS</i>	time-stamp event flag
<i>RTC_FLAG_TSOVR</i>	time-stamp overflow event flag
<i>RTC_FLAG_TP0</i>	RTC tamper 0 detected flag
<i>RTC_FLAG_TP1</i>	RTC tamper 1 detected flag
<i>RTC_FLAG_TP2</i>	RTC tamper 2 detected flag
<i>RTC_FLAG_ITS</i>	internal timestamp flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
FlagStatus	SET or RESET

Example:

```
/* check time-stamp event flag */
```

```
FlagStatus = rtc_flag_get(RTC_FLAG_TS);
```

### rtc\_flag\_clear

The description of rtc\_flag\_clear is shown as below:

**Table 3-940. Function rtc\_flag\_clear**

<b>Function name</b>	rtc_flag_clear
<b>Function prototype</b>	void rtc_flag_clear(uint32_t flag);
<b>Function descriptions</b>	clear specified flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	specify which flag to clear
<i>RTC_FLAG_ALARM0</i> <i>W</i>	Alarm0 writen available flag
<i>RTC_FLAG_ALARM1</i> <i>W</i>	Alarm1 writen available flag
<i>RTC_FLAG_WTW</i>	wakeup timer can be written flag
<i>RTC_FLAG_SOP</i>	shift function operation pending flag
<i>RTC_FLAG_YCM</i>	year parameter configured event flag
<i>RTC_FLAG_RSYN</i>	time and date registers synchronized event flag
<i>RTC_FLAG_INIT</i>	init mode event flag
<i>RTC_FLAG_SCP</i>	smooth calibration pending flag
<i>RTC_FLAG_ALARM0</i>	Alarm0 event flag
<i>RTC_FLAG_ALARM1</i>	Alarm1 event flag
<i>RTC_FLAG_WT</i>	wakeup timer occurs flag
<i>RTC_FLAG_TS</i>	time-stamp event flag
<i>RTC_FLAG_TSOVR</i>	time-stamp overflow event flag

<i>RTC_FLAG_TP0</i>	RTC tamper 0 detected flag
<i>RTC_FLAG_TP1</i>	RTC tamper 1 detected flag
<i>RTC_FLAG_TP2</i>	RTC tamper 2 detected flag
<i>RTC_FLAG_ITS</i>	internal timestamp flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear time-stamp event flag */
rtc_flag_clear(RTC_FLAG_TS);
```

## 3.27. HRTIMER

HRTIMER has a high-precision counting clock and can be used for high-precision timing. It can generate 16 high precision and flexible digital signals to control motor or be used for power management applications. The 16 digital signals can be output independently or coupled into 8 pairs of complementary signals. The HRTIMER registers are listed in chapter [3.27.1](#), the HRTIMER firmware functions are introduced in chapter [3.27.2](#).

### 3.27.1. Descriptions of Peripheral registers

HRTIMER registers are listed in the table shown as below:

**Table 3-941. HRTIMER Register**

Registers	Descriptions
<b>Master Timer Registers</b>	
HRTIMER_MTCTL0	Master_TIMER control register 0
HRTIMER_MTINTF	Master_TIMER interrupt flag register
HRTIMER_MTINTC	Master_TIMER interrupt flag clear register
HRTIMER_MTDMAINTEN	Master_TIMER DMA and interrupt enable register
HRTIMER_MTCNT	Master_TIMER counter register
HRTIMER_MTCAR	Master_TIMER counter auto reload register
HRTIMER_MTCREP	Master_TIMER counter repetition register
HRTIMER_MTCMP0V	Master_TIMER compare 0 value register
HRTIMER_MTCMP1V	Master_TIMER compare 1 value register
HRTIMER_MTCMP2V	Master_TIMER compare 2 value register
HRTIMER_MTCMP3V	Master_TIMER compare 3 value register
<b>Slave Timer Registers</b>	
HRTIMER_STXCTL0	Slave_TIMERx control register 0
HRTIMER_STXINTF	Slave_TIMERx interrupt flag register



Registers	Descriptions
HRTIMER_STXINTC	Slave_TIMERx interrupt flag clear register
HRTIMER_STXDMAINTEN	Slave_TIMERx DMA and interrupt enable register
HRTIMER_STXCNT	Slave_TIMERx counter register
HRTIMER_STXCAR	Slave_TIMERx counter auto reload register
HRTIMER_STXCREP	Slave_TIMERx counter repetition register
HRTIMER_STXCMP0V	Slave_TIMERx compare 0 value register
HRTIMER_STXCMP0CP	Slave_TIMERx compare 0 composite register
HRTIMER_STXCMP1V	Slave_TIMERx compare 1 value register
HRTIMER_STXCMP2V	Slave_TIMERx compare 2 value register
HRTIMER_STXCMP3V	Slave_TIMERx compare 3 value register
HRTIMER_STXCAP0V	Slave_TIMERx capture 0 value register
HRTIMER_STXCAP1V	Slave_TIMERx capture 1 value register
HRTIMER_STXDTCTL	Slave_TIMERx dead-time control register
HRTIMER_STXCH0SET	Slave_TIMERx channel 0 set request register
HRTIMER_STXCH0RST	Slave_TIMERx channel 0 reset request register
HRTIMER_STXCH1SET	Slave_TIMERx channel 1 set request register
HRTIMER_STXCH1RST	Slave_TIMERx channel 1 reset request register
HRTIMER_STXEXEVFCFG0	Slave_TIMERx external event filter configuration register 0
HRTIMER_STXEXEVFCFG1	Slave_TIMERx external event filter configuration register 1
HRTIMER_STXCNTRST	Slave_TIMERx counter reset register
HRTIMER_STXCSCCTL	Slave_TIMERx carrier-signal control register
HRTIMER_STXCAP0TRG	Slave_TIMERx capture 0 trigger register
HRTIMER_STXCAP1TRG	Slave_TIMERx capture 1 trigger register
HRTIMER_STXCHOCTL	Slave_TIMERx channel output control register
HRTIMER_STXFLTCTL	Slave_TIMERx fault control register
HRTIMER_STXCTL1	Slave_TIMERx control register 1
HRTIMER_STXEXEVFCFG2	Slave_TIMERx external event filter configuration register 2
HRTIMER_STXCAPTRGCOM	Slave_TIMERx capture trigger combination register
HRTIMER_STXCNTRSTA	Slave_TIMERx counter reset and channel 0/1 set/reset request add register
HRTIMER_STXACTL	Slave_TIMERx additional control register
<b>Common Registers</b>	
HRTIMER_CTL0	HRTIMER control register 0
HRTIMER_CTL1	HRTIMER control register 1
HRTIMER_INTF	HRTIMER interrupt flag register
HRTIMER_INTC	HRTIMER interrupt flag clear register
HRTIMER_INTEN	HRTIMER interrupt enable register
HRTIMER_CHOUTEN	HRTIMER channel output enable register
HRTIMER_CHOUTDIS	HRTIMER channel output disable register
HRTIMER_CHOUTDISF	HRTIMER channel output disable flag register

Registers	Descriptions
HRTIMER_BMCTL	HRTIMER bunch mode control register
HRTIMER_BMSTRG	HRTIMER bunch mode start trigger register
HRTIMER_BMCMPV	HRTIMER bunch mode compare value register
HRTIMER_BMCAR	HRTIMER bunch mode counter auto reload register
HRTIMER_EXEVCFG0	HRTIMER external event configuration register 0
HRTIMER_EXEVCFG1	HRTIMER external event configuration register 1
HRTIMER_EXEVDCTL	HRTIMER external event digital filter control register
HRTIMER_ADCTRIGS0	HRTIMER trigger source 0 to ADC register
HRTIMER_ADCTRIGS1	HRTIMER trigger source 1 to ADC register
HRTIMER_ADCTRIGS2	HRTIMER trigger source 2 to ADC register
HRTIMER_ADCTRIGS3	HRTIMER trigger source 3 to ADC register
HRTIMER_DLLCCTL	HRTIMER DLL calibration control register
HRTIMER_FLTINCFG0	HRTIMER fault input configuration register 0
HRTIMER_FLTINCFG1	HRTIMER fault input configuration register 1
HRTIMER_DMAUPMTR	HRTIMER DMA update Master_TIMER register
HRTIMER_DMAUPST0R	HRTIMER DMA update Slave_TIMER0 register
HRTIMER_DMAUPST1R	HRTIMER DMA update Slave_TIMER1 register
HRTIMER_DMAUPST2R	HRTIMER DMA update Slave_TIMER2 register
HRTIMER_DMAUPST3R	HRTIMER DMA update Slave_TIMER3 register
HRTIMER_DMAUPST4R	HRTIMER DMA update Slave_TIMER4 register
HRTIMER_DMAUPST5R	HRTIMER DMA update Slave_TIMER5 register
HRTIMER_DMAUPST6R	HRTIMER DMA update Slave_TIMER6 register
HRTIMER_DMAUPST7R	HRTIMER DMA update Slave_TIMER7 register
HRTIMER_DMATB	HRTIMER DMA transfer buffer register
HRTIMER_ADCEXTTRG	HRTIMER ADC extended trigger register
HRTIMER_ADCTRIGUPD	HRTIMER ADC trigger update register
HRTIMER_ADCPSCR0	HRTIMER ADC prescaler register 0
HRTIMER_ADCPSCR1	HRTIMER ADC prescaler register 1
HRTIMER_FLTINCFG2	HRTIMER fault input configuration register 2
HRTIMER_FLTINCFG3	HRTIMER fault input configuration register 3
HRTIMER_BMSTRGA	HRTIMER bunch mode start trigger add register
HRTIMER_FLTINCFG4	HRTIMER fault input configuration register 4
HRTIMER_ADCEXTTRGA	HRTIMER ADC extended trigger add register
HRTIMER_ADCTRIGS0A	HRTIMER trigger source 0 to ADC add register
HRTIMER_ADCTRIGS1A	HRTIMER trigger source 1 to ADC add register
HRTIMER_ADCTRIGS2A	HRTIMER trigger source 2 to ADC add register
HRTIMER_ADCTRIGS3A	HRTIMER trigger source 3 to ADC add register

### 3.27.2. Descriptions of Peripheral functions

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

Table 3-942. HRTIMER firmware function

Function name	Function description
hrtimer_deinit	deinit a HRTIMER
hrtimer_dll_calibration_start	configure and start DLL calibration
hrtimer_baseinit_struct_para_init	initialize HRTIMER time base parameters struct with a default value
hrtimer_timers_base_init	initialize Master_TIMER and Slave_TIMER timerbase
hrtimer_timers_counter_enable	enable a counter
hrtimer_timers_counter_disable	disable a counter
hrtimer_timers_update_event_enable	enable the Master_TIMER or Slave_TIMER update event
hrtimer_timers_update_event_disable	disable the Master_TIMER or Slave_TIMER update event
hrtimer_software_update	trigger the Master_TIMER and Slave_TIMER registers update by software
hrtimer_software_counter_reset	the Master_TIMER and Slave_TIMER counter reset by software
hrtimer_output_exchange	exchange Slave_TIMERx outputs
hrtimer_timerinit_struct_para_init	initialize waveform mode initialization parameters struct with a default value
hrtimer_timers_waveform_init	initialize a timer to work in waveform mode
hrtimer_timercfg_struct_para_init	initialize Slave_TIMER general behavior configuration struct with a default value
hrtimer_slavetimer_waveform_config	configure the general behavior of a Slave_TIMER which work in waveform mode
hrtimer_comparecfg_struct_para_init	initialize compare unit configuration struct with a default value
hrtimer_slavetimer_waveform_compare_config	configure the compare unit of a Slave_TIMER which work in waveform mode
hrtimer_channel_outputcfg_struct_para_init	initialize channel output configuration struct with a default value
hrtimer_slavetimer_waveform_channel_config	configure the channel output of a Slave_TIMER work in waveform mode
hrtimer_slavetimer_waveform_channel_software_request	software generates channel "set request" or "reset request"
hrtimer_slavetimer_waveform_channel_output_level_get	get Slave_TIMER channel output level
hrtimer_slavetimer_waveform_channel_state_get	get Slave_TIMER channel run state
hrtimer_deadtimercfg_struct_para_init	initialize dead time configuration struct with a default value
hrtimer_slavetimer_deadtime_config	configure the dead time for Slave_TIMER
hrtimer_carriersignalcfg_struct_para_init	initialize carrier signal configuration struct with a default value
hrtimer_slavetimer_carriersignal_config	configure the carrier signal mode for Slave_TIMER

Function name	Function description
hrtimer_output_channel_enable	enable a output channel
hrtimer_output_channel_disable	disable a output channel
hrtimer_mastertimer_compare_value_config	configure the compare value in Master_TIMER
hrtimer_mastertimer_compare_value_get	get the compare value in Master_TIMER
hrtimer_slavetimer_compare_value_config	configure the compare value in Slave_TIMER
hrtimer_slavetimer_compare_value_get	get the compare value in Slave_TIMER
hrtimer_timers_counter_value_config	configure the counter value in Master_TIMER and Slave_TIMER
hrtimer_timers_counter_value_get	get the counter value in Master_TIMER and Slave_TIMER
hrtimer_timers_autoreload_value_config	configure the counter auto reload value in Master_TIMER and Slave_TIMER
hrtimer_timers_autoreload_value_get	get the counter auto reload value in Master_TIMER and Slave_TIMER
hrtimer_timers_repetition_value_config	configure the counter repetition value in Master_TIMER and Slave_TIMER
hrtimer_timers_repetition_value_get	get the counter repetition value in Master_TIMER and Slave_TIMER
hrtimer_exefilter_struct_para_init	initialize external event filtering for Slave_TIMER configuration struct with a default value
hrtimer_slavetimer_exeevent_filtering_config	configure the external event filtering for Slave_TIMER (blanking, windowing)
hrtimer_exeeventcfg_struct_para_init	initialize external event configuration struct with a default value
hrtimer_exeevent_config	configure the an external event
hrtimer_exeevent_prescaler	configure external event digital filter clock division
hrtimer_exeeventx_counter_struct_para_init	initialize external event X counter configuration struct with a default value
hrtimer_exeeventx_counter_config	configure the external event X counter
hrtimer_software_reset_exeeventx_counter	reset external event X counter by software
hrtimer_exeeventx_counter_enable	enable external event X counter
hrtimer_exeeventx_counter_disable	disable external event X counter
hrtimer_exeeventx_counter_read	read external event X counter value
hrtimer_synccfg_struct_para_init	initialize synchronization configuration struct with a default value
hrtimer_synchronization_config	configure the synchronization input/output of the HRTIMER
hrtimer_double_channel_struct_para_init	initialize double channel trigger struct

Function name	Function description
hrtimer_double_trigger_config	configure Slave_TIMER in double channel trigger
hrtimer_roll_over_struct_para_init	initialize roll over struct with a default value
hrtimer_roll_over_mode_config	configure Slave_TIMER in roll over mode
hrtimer_faultcfg_struct_para_init	configure the synchronization input/output of the HRTIMER
hrtimer_fault_config	configure the fault input
hrtimer_fault_prescaler_config	configure the fault input digital filter clock division
hrtimer_fault_input_enable	fault input enable
hrtimer_fault_input_disable	fault input disable
hrtimer_fault_counter_reset	fault input counter reset
hrtimer_fault_blank_enable	fault input blank enable
hrtimer_fault_blank_disable	fault input blank disable
hrtimer_timers_dma_enable	enable the Master_TIMER and Slave_TIMER DMA request
hrtimer_timers_dma_disable	disable the Master_TIMER and Slave_TIMER DMA request
hrtimer_dmamode_config	configure the DMA mode for Master_TIMER or Slave_TIMER
hrtimer_bunchmode_struct_para_init	initialize bunch mode configuration struct with a default value
hrtimer_bunchmode_config	configure bunch mode for the HRTIMER
hrtimer_bunchmode_enable	enable the bunch mode
hrtimer_bunchmode_disable	disable the bunch mode
hrtimer_bunchmode_flag_get	get bunch mode operating flag
hrtimer_bunchmode_software_start	bunch mode started by software
hrtimer_slavetimer_capture_config	configure the capture source in Slave_TIMER
hrtimer_slavetimer_capture_software	capture triggered by software in Slave_TIMER
hrtimer_slavetimer_capture_value_read	read the capture value
hrtimer_adctrigcfg_struct_para_init	initialize ADC trigger configuration struct with a default value
hrtimer_adc_trigger0_3_config	configure the trigger source to ADC and the update source(0..3)
hrtimer_adc_trigger4_9_config	configure the trigger source to ADC and the update source(4..9)
hrtimer_adc_prescaler_config	configure the trigger source prescaler of ADC (0..9)
hrtimer_slavetimer_counter_direction_get	get the Slave_TIMER counter direction
hrtimer_timers_flag_get	get the Master_TIMER and Slave_TIMER flag
hrtimer_timers_flag_clear	clear the Master_TIMER and Slave_TIMER flag
hrtimer_common_flag_get	get the common flag
hrtimer_common_flag_clear	clear the common flag
hrtimer_timers_interrupt_enable	enable the Master_TIMER and Slave_TIMER interrupt
hrtimer_timers_interrupt_disable	disable the Master_TIMER and Slave_TIMER interrupt
hrtimer_timers_interrupt_flag_get	clear the Master_TIMER and Slave_TIMER interrupt flag
hrtimer_timers_interrupt_flag_clear	clear the Master_TIMER and Slave_TIMER interrupt flag
hrtimer_common_interrupt_enable	enable the common interrupt

Function name	Function description
hrtimer_common_interrupt_disable	disable common interrupt
hrtimer_common_interrupt_flag_get	clear the common interrupt flag
hrtimer_common_interrupt_flag_clear	clear the common interrupt flag

### Structure hrtimer\_baseinit\_parameter\_struct

**Table 3-943. Structure hrtimer\_baseinit\_parameter\_struct**

member name	Function description
period	period value, min value: 3 tHRTIMER_CK clock, max value: 0xFFFF - (1 tHRTIMER_CK)
repetitioncounter	the counter repetition value, 0x00~0xFF
prescaler	prescaler value
counter_mode	counter operating mode
counterdirection	counter direction

### Structure hrtimer\_timerinit\_parameter\_struct

**Table 3-944. Structure hrtimer\_timerinit\_parameter\_struct**

member name	Function description
half_mode	specifies whether or not half mode is enabled, refer to: half mode enabling status
alternate_mode	specifies whether or not alternate mode is enabled, refer to: alternate mode enable status
resynchronized_update	specifies whether or not the update source is coming from other slave timer
start_sync	specifies whether or not timer is started by a rising edge on the synchronization input, refer to: synchronous input start timer
reset_sync	specifies whether or not timer is reset by a rising edge on the synchronization input, refer to: synchronous input reset timer
dac_trigger	indicates whether or not the a DAC synchronization event is generated, refer to: trigger source to DAC
shadow	Indicates whether or not the a DAC synchronization event is generated, refer to: trigger source to DAC
update_selection	the update occurs with respect to DMA mode or STxUPINy (Slave_TIMERx only), refer to: update event selection
cnt_bunch	the timer behaves during a bunch mode operation, refer to: timer behaves during a bunch mode operation
repetition_update	the timer behaves during a bunch mode operation, refer to: timer behaves during a bunch mode operation

## Structure `hrtimer_timercfg_parameter_struct`

**Table 3-945. Structure `hrtimer_timercfg_parameter_struct`**

member name	Function description
<code>balanced_mode</code>	specifies whether or not the balanced mode is enabled, refer to: set balanced mode
<code>fault_enable</code>	specifies whether or not the fault channels are enabled for the Slave_TIMER, refer to: fault channel enabled for a Slave_TIMER
<code>fault_protect</code>	specifies whether the write protection function is enable or not, refer to: protect fault enable
<code>fault_automatic_resume</code>	specifies whether the write fault automatic resume is enable or not, refer to: fault auto resume
<code>deadtime_enable</code>	specifies whether or not dead time insertion is enabled for the timer, refer to: dead time enable
<code>delayed_idle</code>	the delayed IDLE mode, refer to: set delayed IDLE state mode
<code>balanced_idle_automatic_resume</code>	specifies whether or not the balanced idle automatic resume is enabled, refer to: balanced idle automatic resume enable
<code>update_source</code>	the source triggering the Slave_TIMER registers update, refer to: update is done synchronously with any other Slave_TIMER or Master_TIMER update
<code>cnt_reset</code>	the source triggering the Slave_TIMER counter reset, refer to: Slave_TIMER counter reset
<code>reset_update</code>	specifies whether or not registers update is triggered when the timer counter is reset, refer to: update event generated by reset event

## Structure `hrtimer_capture_value_struct`

**Table 3-946. Structure `hrtimer_capture_value_struct`**

member name	Function description
<code>value</code>	capture value
<code>dir</code>	count direction when captured

## Structure `hrtimer_comparecfg_parameter_struct`

**Table 3-947. Structure `hrtimer_comparecfg_parameter_struct`**

member name	Function description
<code>compare_value</code>	compare value, min value: 3 tHRTIMER_CK clock, max value: 0xFFFF - (1 tHRTIMER_CK)
<code>delayed_mode</code>	defining whether the compare register is behaving in regular mode or in delayed mode, refer to: compare 3 or 1 delayed mode
<code>timeout_value</code>	compare value for compare 0 or 2 when compare 3 or 1 is delayed mode with time out is selected, <code>timeout_value + compare_value</code> must be less than 0xFFFF
<code>trigger_half</code>	triggered half mode
<code>immediately_update</code>	immediately update compare 0 value PWM mode

_cmp0	
immediately_update _cmp2	immediately update compare 2 value PWM mode

### Structure hrtimer\_exevfiter\_parameter\_struct

**Table 3-948. Structure hrtimer\_exevfiter\_parameter\_struct**

member name	Function description
filter_mode	the external event filter mode for Slave_TIMER, refer to: external event filter mode
memorized	specifies whether or not the signal is memorized, refer to: external event memorized enable

### Structure hrtimer\_deadtimecfg\_parameter\_struct

**Table 3-949. Structure hrtimer\_deadtimecfg\_parameter\_struct**

member name	Function description
prescaler	dead time generator clock division, refer to: dead time prescaler
rising_value	rising edge dead-time value, 0x0000~0xFFFF
rising_sign	the sign of rising edge dead-time value, refer to: dead time rising sign
rising_protect	dead time rising edge protection for value and sign, refer to: dead time rising edge protection for value and sign
risingsign_protect	dead time rising edge protection for sign, refer to: dead time rising edge protection only for sign
falling_value	falling edge dead-time value, 0x0000~0xFFFF
falling_sign	the sign of falling edge dead-time value, refer to: dead time falling sign
falling_protect	dead time falling edge protection for value and sign, refer to: dead time falling edge protection for value and sign
fallingsign_protect	dead time falling edge protection for sign, refer to: dead time falling edge protection only for sign

### Structure hrtimer\_carriersignalcfg\_parameter\_struct

**Table 3-950. Structure hrtimer\_carriersignalcfg\_parameter\_struct**

member name	Function description
period	carrier signal period: tCSPRD, 0x0~0xF. $tCSPRD = (period + 1) * 16 * tHRTIMER\_CK$
duty_cycle	carrier signal duty cycle, 0x0~0x7, duty cycle = duty_cycle/8
first_pulse	first carrier-signal pulse width: tCSFSTPW, 0x0~0xF. $tCSFSTPW = (first\_pulse+1) * 16 * tHRTIMER\_CK$



## Structure `hrtimer_synccfg_parameter_struct`

**Table 3-951. Structure `hrtimer_synccfg_parameter_struct`**

member name	Function description
input_source	the external synchronization input source, refer to: the synchronization input source
output_source	the source and event to be sent on the external synchronization outputs, refer to: the synchronization output source
output_polarity	the polarity and length of the pulse to be sent on the external synchronization outputs, refer to: the pulse on the synchronization output pad HRTIMER_SCOUT

## Structure `hrtimer_bunchmode_parameter_struct`

**Table 3-952. Structure `hrtimer_bunchmode_parameter_struct`**

member name	Function description
mode	the bunch mode operating mode, refer to: continuous mode in bunch mode
clock_source	specifies the burst mode clock source, refer to: bunch mode clock source
prescaler	the bunch mode prescaler, refer to: bunch mode clock division
shadow	specifies whether or not preload is enabled for HRTIMER_BMCMPV and HRTIMER_BMCAR registers, refer to: bunch mode shadow enable
Trigger[2]	the event triggering the bunch operation, refer to: the event triggers bunch mode operation
idle_duration	the duration of the IDLE, 0x0000~0xFFFF
period	the bunch mode period which is the sum of the IDLE and RUN duration, 0x0001~0xFFFF

## Structure `hrtimer_exeventcfg_parameter_struct`

**Table 3-953. Structure `hrtimer_exeventcfg_parameter_struct`**

member name	Function description
source	the source of the external event, refer to: external event source
polarity	the active level of external event 0 when EXEVyEG[1:0] = 2'b00, refer to: external event polarity
edge	the sensitivity of the external event, external event edge sensitivity
fast	external event fast mode
digital_filter	external event filter control, 0x0~0xF

## Structure `hrtimer_exeventcnt_parameter_struct`

**Table 3-954. Structure `hrtimer_exeventcnt_parameter_struct`**

member name	Function description
reset_mode	external event X reset mode
counter_threshold	external event X counter threshold value , 0x00~0x3F

event_source	external event X selection
--------------	----------------------------

### Structure `hrtimer_faultcfg_parameter_struct`

**Table 3-955. Structure `hrtimer_faultcfg_parameter_struct`**

member name	Function description
source	the source of the fault input, refer to: fault input source
polarity	the polarity of the fault input, refer to: fault input polarity
filter	fault input filter control, 0x0~0xF
control	fault input enable or disable, refer to: enable or disable fault
protect	protect fault input configuration, refer to: protect fault input configuration
blanksource	fault blanking source
counter	fault blanking counter
resetmode	fault reset mode

### Structure `hrtimer_adctrigcfg_parameter_struct`

**Table 3-956. Structure `hrtimer_adctrigcfg_parameter_struct`**

member name	Function description
update_source	the source triggering the update of the HRTIMER_ADCTRIGSy register, refer to: HRTIMER_ADCTRIG update source
Trigger0_3[2]	the event triggering the ADC conversion, refer to: ADC trigger 0,2 and ADC trigger 1,3
trigger4_9	the event triggering the ADC conversion, refer to: ADC trigger 4,6,8 and ADC trigger 5,7,9

### Structure `hrtimer_channel_outputcfg_parameter_struct`

**Table 3-957. Structure `hrtimer_channel_outputcfg_parameter_struct`**

member name	Function description
polarity	configure channel output polarity, refer to: channel output polarity
set_request	configure the event generates channel 'set request', refer to channel set request
reset_request	configure the event generates channel 'reset request', refer to: channel reset request
idle_bunch	specifies whether channel output can be IDLE state in bunch mode, refer to: channel IDLE state enable in bunch mode
idle_state	specifies channel output idle state, refer to channel output idle state
fault_state	specifies the output level when in FAULT state, refer to: channel output in fault state
carrier_mode	specifies whether or not the carrier-signal mode is enabled, refer to: channel carrier-signal mode enable
deadtime_bunch	specifies whether or not deadtime is inserted before output entering the IDLE state in bunch mode, refer to: channel dead-time insert in bunch mode

## Structure `hrtimer_roll_over_parameter_struct`

**Table 3-958. Structure `hrtimer_roll_over_parameter_struct`**

member name	Function description
<code>roll_over_mode</code>	roll-over mode
<code>output_roll_over_mode</code>	output roll-over mode
<code>adc_roll_over_mode</code>	ADC roll-over mode
<code>bunch_mode_roll_over_mode</code>	bunch mode roll-over mode
<code>fault_event_roll_over_mode</code>	fault and event roll-over mode

## Structure `hrtimer_double_trigger_parameter_struct`

**Table 3-959. Structure `hrtimer_double_trigger_parameter_struct`**

member name	Function description
<code>trigger_enable</code>	configure the double trigger, refer to: double trigger enable
<code>trigger0</code>	configure trigger0
<code>trigger1</code>	configure trigger1

## `hrtimer_deinit`

The description of `hrtimer_deinit` is shown as below:

**Table 3-960. Function `hrtimer_deinit`**

<b>Function name</b>	<code>hrtimer_deinit</code>
<b>Function prototype</b>	<code>void hrtimer_deinit(uint32_t hrtimer_periph);</code>
<b>Function descriptions</b>	deinit HRTIMER
<b>Precondition</b>	-
<b>The called functions</b>	<code>rcu_periph_reset_enable</code> / <code>rcu_periph_reset_disable</code>
<b>Input parameter{in}</b>	
<b><code>hrtimer_periph</code></b>	HRTIMER peripheral
<b><code>HRTIMERx(x=0)</code></b>	HRTIMER selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* reset HRTIMER */
hrtimer_deinit(HRTIMER0);

```

## hrtimer\_dll\_calibration\_start

The description of hrtimer\_dll\_calibration\_start is shown as below:

**Table 3-961. Function hrtimer\_dll\_calibration\_start**

<b>Function name</b>	hrtimer_dll_calibration_start
<b>Function prototype</b>	void hrtimer_dll_calibration_start(uint32_t hrtimer_periph);
<b>Function descriptions</b>	configure and start DLL calibration
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>calform</b>	specify the calibration form
<i>HRTIMER_CALIBRATION_ONCE</i>	DLL calibration start once
<i>HRTIMER_CALIBRATION_1048576_PERIOD</i>	DLL periodic calibration, the length of the DLL calibration cycle is 1048576 * tHRTIMER_CK
<i>HRTIMER_CALIBRATION_131072_PERIOD</i>	DLL periodic calibration, the length of the DLL calibration cycle is 131072 * tHRTIMER_CK
<i>HRTIMER_CALIBRATION_16384_PERIOD</i>	DLL periodic calibration, the length of the DLL calibration cycle is 16384 * tHRTIMER_CK
<i>HRTIMER_CALIBRATION_2048_PERIOD</i>	DLL periodic calibration, the length of the DLL calibration cycle is 2048 * tHRTIMER_CK
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure and start DLL calibration */
```

```
hrtimer_dll_calibration_start(HRTIMER0, HRTIMER_CALIBRATION_ONCE);
```

## hrtimer\_baseinit\_struct\_para\_init

The description of hrtimer\_baseinit\_struct\_para\_init is shown as below:

**Table 3-962. Function hrtimer\_baseinit\_struct\_para\_init**

<b>Function name</b>	hrtimer_baseinit_struct_para_init
<b>Function prototype</b>	void hrtimer_baseinit_struct_para_init(hrtimer_baseinit_parameter_struct* baseinit);
<b>Function descriptions</b>	initialize HRTIMER time base parameters struct with the default value
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
baseinit	hrtimer_baseinit_parameter_struct, the structure members can refer to <a href="#">Table 3-943. Structure hrtimer_baseinit_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize HRTIMER time base parameters struct with the default value */
```

```
hrtimer_baseinit_parameter_struct baseinit;
```

```
hrtimer_baseinit_struct_para_init(&baseinit);
```

### hrtimer\_timers\_base\_init

The description of hrtimer\_timers\_base\_init is shown as below:

**Table 3-963. Function hrtimer\_timers\_base\_init**

Function name	hrtimer_timers_base_init
Function prototype	void hrtimer_timers_base_init(uint32_t hrtimer_periph);
Function descriptions	initialize Master_TIMER and Slave_TIMER timerbase
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
timer_id	master timer and slave timer index
HRTIMER_MASTER_TIMER	the master timer
HRTIMER_SLAVE_TIMERx	the slave timer selection(x=0.. <b>7</b> )
Input parameter{in}	
baseinit	hrtimer_baseinit_parameter_struct, the structure members can refer to <a href="#">Table 3-943. Structure hrtimer_baseinit_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize Master_TIMER and Slave_TIMER timerbase */
```

```

hrtimer_baseinit_parameter_struct baseinit_para;

hrtimer_baseinit_struct_para_init(&baseinit_para);

baseinit_para.period = 384;

baseinit_para.prescaler = HRTIMER_PRESCALER_MUL32;

baseinit_para.repetitioncounter = 0;

baseinit_para.Counterdirection = HRTIMER_COUNTER_UP;

baseinit_para.counter_mode = HRTIMER_COUNTER_MODE_CONTINUOUS;

hrtimer_timers_base_init(HRTIMER0, HRTIMER_SLAVE_TIMER0, &baseinit_para);

```

### hrtimer\_timers\_counter\_enable

The description of hrtimer\_timers\_counter\_enable is shown as below:

**Table 3-964. Function hrtimer\_timers\_counter\_enable**

<b>Function name</b>	hrtimer_timers_counter_enable
<b>Function prototype</b>	void hrtimer_timers_counter_enable(uint32_t hrtimer_periph, uint32_t cntid);
<b>Function descriptions</b>	enable a counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>cntid</b>	specify the counter to configure
<i>HRTIMER_MT_COUNTER</i>	the counter of Master_TIMER
<i>HRTIMER_STx_COUNTER(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* enable a counter */

void hrtimer_timers_counter_enable(HRTIMER0, HRTIMER_MT_COUNTER);

```

### hrtimer\_timers\_counter\_disable

The description of hrtimer\_timers\_counter\_disable is shown as below:

Table 3-965. Function `hrtimer_timers_counter_disable`

<b>Function name</b>	<code>hrtimer_timers_counter_disable</code>
<b>Function prototype</b>	<code>void hrtimer_timers_counter_disable(uint32_t hrtimer_periph, uint32_t cntid);</code>
<b>Function descriptions</b>	disable a counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>cntid</b>	specify the counter to configure
<i>HRTIMER_MT_COUNTER</i>	the counter of Master_TIMER
<i>HRTIMER_STx_COUNTER(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable a counter */
```

```
hrtimer_timers_counter_disable(HRTIMER0, HRTIMER_MT_COUNTER);
```

### **`hrtimer_timers_update_event_enable`**

The description of `hrtimer_timers_update_event_enable` is shown as below:

Table 3-966. Function `hrtimer_timers_update_event_enable`

<b>Function name</b>	<code>hrtimer_timers_update_event_enable</code>
<b>Function prototype</b>	<code>void hrtimer_timers_update_event_enable(uint32_t hrtimer_periph, uint32_t timer_id);</code>
<b>Function descriptions</b>	enable the Master_TIMER or Slave_TIMER update event
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_MASTER_TIMER</i>	the counter of Master_TIMER

<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the Master_TIMER or Slave_TIMER update event */
```

```
hrtimer_timers_update_event_enable(HRTIMER0, HRTIMER_MASTER_TIMER);
```

### **hrtimer\_timers\_update\_event\_disable**

The description of hrtimer\_timers\_update\_event\_disable is shown as below:

**Table 3-967. Function hrtimer\_timers\_update\_event\_enable**

<b>Function name</b>	hrtimer_timers_update_event_disable
<b>Function prototype</b>	void hrtimer_timers_update_event_disable(uint32_t hrtimer_periph, uint32_t timer_id);
<b>Function descriptions</b>	disable the Master_TIMER or Slave_TIMER update event
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_MASTER_TIMER</i>	the counter of Master_TIMER
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the Master_TIMER or Slave_TIMER update event */
```

```
hrtimer_timers_update_event_disable(HRTIMER0, HRTIMER_MASTER_TIMER);
```

### **hrtimer\_software\_update**

The description of hrtimer\_software\_update is shown as below:



Table 3-968. Function `hrtimer_software_update`

<b>Function name</b>	<code>hrtimer_software_update</code>
<b>Function prototype</b>	<code>void hrtimer_software_update(uint32_t hrtimer_periph, uint32_t timersrc);</code>
<b>Function descriptions</b>	update the Master_TIMER or Slave_TIMER by software
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timersrc</b>	software update timer selection
<i>HRTIMER_UPDATE_SW_MT</i>	Master_TIMER software update
<i>HRTIMER_UPDATE_SW_STx(x=0..7)</i>	Slave_TIMERx software update (x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* update the Master_TIMER or Slave_TIMER by software */
```

```
void hrtimer_software_update(HRTIMER0, HRTIMER_UPDATE_SW_MT);
```

### **hrtimer\_software\_counter\_reset**

The description of `hrtimer_software_counter_reset` is shown as below:

Table 3-969. Function `hrtimer_software_counter_reset`

<b>Function name</b>	<code>hrtimer_software_counter_reset</code>
<b>Function prototype</b>	<code>void hrtimer_software_counter_reset(uint32_t hrtimer_periph, uint32_t timerrst);</code>
<b>Function descriptions</b>	reset the Master_TIMER or Slave_TIMER counter by software
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timersrc</b>	software reset timer selection
<i>HRTIMER_COUNTER_RESET_SW_MT</i>	Master_TIMER software reset
<i>HRTIMER_COUNTER_RESET_SW_STx(x=0..7)</i>	Slave_TIMERx software reset(x=0..7)

<i>RESET_SW_STx</i> ( <i>x=0..7</i> )	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* triggers the Master_TIMER or Slave_TIMER registers update by software */
```

```
void hrtimer_software_counter_reset(HRTIMER0, HRTIMER_COUNTER_RESET_SW_MT);
```

### hrtimer\_output\_exchange

The description of hrtimer\_output\_exchange is shown as below:

**Table 3-970. Function hrtimer\_output\_exchange**

<b>Function name</b>	hrtimer_output_exchange
<b>Function prototype</b>	void hrtimer_output_exchange(uint32_t hrtimer_periph, uint32_t timerexc);
<b>Function descriptions</b>	exchange Slave_TIMERx outputs
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
Input parameter{in}	
<b>timersrc</b>	output exchange Slave_TIMERx
<i>HRTIMER_OUTPUT_EXCHANGE_STx</i> ( <i>x=0..7</i> )	Slave_TIMERx exchange output ( <i>x=0..7</i> )
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* exchange Slave_TIMER0 outputs */
```

```
void hrtimer_output_exchange(HRTIMER0, HRTIMER_HRTIMER_OUTPUT_EXCHANGE_ST0);
```

### hrtimer\_timerinit\_struct\_para\_init

The description of hrtimer\_timerinit\_struct\_para\_init is shown as below:

Table 3-971. Function `hrtimer_timerinit_struct_para_init`

Function name	<code>hrtimer_timerinit_struct_para_init</code>
Function prototype	<code>void hrtimer_timerinit_struct_para_init(hrtimer_timerinit_parameter_struct* timerinit);</code>
Function descriptions	initialize waveform mode initialization parameters struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
timerinit	<code>hrtimer_timerinit_parameter_struct</code> , the structure members can refer to <a href="#">Table 3-944. Structure <code>hrtimer_timerinit_parameter_struct</code></a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize waveform mode initialization parameters struct with a default value */
hrtimer_timerinit_parameter_struct timerinit_para;
hrtimer_timerinit_struct_para_init(&timerinit_para);
```

## hrtimer\_timers\_waveform\_init

The description of `hrtimer_timers_waveform_init` is shown as below:

Table 3-972. Function `hrtimer_timers_waveform_init`

Function name	<code>hrtimer_timers_waveform_init</code>
Function prototype	<code>void hrtimer_timers_waveform_init(uint32_t hrtimer_periph, uint32_t timer_id, hrtimer_timerinit_parameter_struct* timerinitpara);</code>
Function descriptions	initialize a timer to work in waveform mode
Precondition	-
The called functions	<code>master_timer_waveform_config</code> / <code>slave_timer_waveform_config</code>
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
<code>HRTIMERx(x=0)</code>	HRTIMER selection
Input parameter{in}	
timer_id	HRTIMER peripheral
<code>HRTIMER_MASTER_TIMER</code>	the counter of Master_TIMER
<code>HRTIMER_SLAVE_TIMERx(x=0..7)</code>	the counter of Slave_TIMERx(x=0..7)
Input parameter{in}	
timerinit	<code>hrtimer_timerinit_parameter_struct</code> , the structure members can refer to <a href="#">Table 3-944. Structure <code>hrtimer_timerinit_parameter_struct</code></a>

Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize a timer to work in waveform mode */

hrtimer_timerinit_parameter_struct timerinit_para;

hrtimer_timerinit_struct_para_init(&timerinit_para);

timerinit_para.cnt_bunch = HRTIMER_TIMERBUNCHMODE_MAINTAINCLOCK;

timerinit_para.DAC_trigger = HRTIMER_DAC_TRIGGER_NONE;

timerinit_para.half_mode = HRTIMER_HALFMODE_DISABLED;

timerinit_para.alternate_mode = HRTIMER_ALTERNATE_MODE_DISABLED;

timerinit_para.repetition_update = HRTIMER_UPDATEONREPETITION_DISABLED;

timerinit_para.reset_sync = HRTIMER_SYNCRESET_DISABLED;

timerinit_para.shadow = HRTIMER_SHADOW_DISABLED;

timerinit_para.start_sync = HRTIMER_SYNISTART_DISABLED;

timerinit_para.update_selection                                =
HRTIMER_MT_ST_UPDATE_SELECTION_INDEPENDENT;

timerinit_para.resynchronized_update = HRTIMER_RSYUPD_DISABLED;

hrtimer_timers_waveform_init(HRTIMER0, HRTIMER_SLAVE_TIMER0, &timerinit_para);

```

### hrtimer\_timercfg\_struct\_para\_init

The description of hrtimer\_timercfg\_struct\_para\_init is shown as below:

**Table 3-973. Function hrtimer\_timercfg\_struct\_para\_init**

Function name	hrtimer_timercfg_struct_para_init
Function prototype	void hrtimer_timercfg_struct_para_init(hrtimer_timercfg_parameter_struct* timercfg);
Function descriptions	initialize Slave_TIMER general behavior configuration struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
timercfg	hrtimer_timercfg_parameter_struct, the structure members can refer to <a href="#">Table 3-945. Structure hrtimer_timercfg_parameter_struct</a>

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize Slave_TIMER general behavior configuration struct with a default value */
hrtimer_timercfg_parameter_struct timercfg_para;
hrtimer_timercfg_struct_para_init(timercfg_para);
```

### hrtimer\_slavetimer\_waveform\_config

The description of hrtimer\_slavetimer\_waveform\_config is shown as below:

**Table 3-974. Function hrtimer\_slavetimer\_waveform\_config**

Function name	hrtimer_slavetimer_waveform_config
Function prototype	void hrtimer_slavetimer_waveform_config(uint32_t hrtimer_periph, uint32_t timer_id, hrtimer_timercfg_parameter_struct * timercfg);
Function descriptions	initialize Slave_TIMER general behavior configuration struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
timer_id	HRTIMER peripheral
HRTIMER_MASTER_TIMER	the counter of Master_TIMER
HRTIMER_SLAVE_TIMERx(x=0..7)	the counter of Slave_TIMERx(x=0..7)
Input parameter{in}	
timercfg	hrtimer_timercfg_parameter_struct, the structure members can refer to <a href="#">Table 3-945. Structure hrtimer_timercfg_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize Slave_TIMER general behavior configuration struct with a default value */
hrtimer_timercfg_parameter_struct timercfg_para;
```

```

hrtimer_timercfg_struct_para_init(&timercfg_para);

timercfg_para.balanced_mode = HRTIMER_STXBALANCEDMODE_DISABLED;

timercfg_para.balanced_idle_automatic_resumption =
HRTIMER_BALANE_IDLE_AUTOMATIC_RESUME_DISABLE;

timercfg_para.cnt_reset = HRTIMER_STXCNT_RESET_NONE;

timercfg_para.deadtime_enable = HRTIMER_STXDEADTIME_DISABLED;

timercfg_para.delayed_idle = HRTIMER_STXDELAYED_IDLE_DISABLED;

timercfg_para.fault_enable = HRTIMER_STXFAULTENABLE_NONE;

timercfg_para.fault_protect = HRTIMER_STXFAULT_PROTECT_READWRITE;

timercfg_para.reset_update = HRTIMER_STXUPDATEONRESET_DISABLED;

timercfg_para.update_source = HRTIMER_STXUPDATETRIGGER_NONE;

hrtimer_slavetimer_waveform_config(HRTIMER0, HRTIMER_SLAVE_TIMER0,
&timercfg_para);

```

### hrtimer\_comparecfg\_struct\_para\_init

The description of hrtimer\_comparecfg\_struct\_para\_init is shown as below:

**Table 3-975. Function hrtimer\_comparecfg\_struct\_para\_init**

Function name	hrtimer_comparecfg_struct_para_init
Function prototype	void hrtimer_comparecfg_struct_para_init(hrtimer_comparecfg_parameter_struct * comparecfg);
Function descriptions	initialize compare unit configuration struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
comparecfg	timer parameter initialization struct, the structure members can refer to <a href="#">Table 3-947. Structure hrtimer_comparecfg_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize compare unit configuration struct with a default value */

hrtimer_comparecfg_parameter_struct comparecfg_para;

hrtimer_comparecfg_struct_para_init(&comparecfg_para);

```

## hrtimer\_slavetimer\_waveform\_compare\_config

The description of hrtimer\_slavetimer\_waveform\_compare\_config is shown as below:

**Table 3-976. Function hrtimer\_slavetimer\_waveform\_compare\_config**

<b>Function name</b>	hrtimer_slavetimer_waveform_compare_config
<b>Function prototype</b>	void hrtimer_slavetimer_waveform_compare_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t comparex, hrtimer_comparecfg_parameter_struct* cmpcfg);
<b>Function descriptions</b>	configure the compare unit of a Slave_TIMER which work in waveform mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	
<b>comparex</b>	compare unit selection
<i>HRTIMER_COMPAREy(y=0..3)</i>	the compare unit y(y=0..3)
<b>Input parameter{in}</b>	
<b>comparecfg</b>	timer parameter initialization struct, the structure members can refer to <a href="#">Table 3-947. Structure hrtimer_comparecfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure the compare unit of a Slave_TIMER which work in waveform mode */

hrtimer_comparecfg_parameter_struct comparecfg_para;

hrtimer_comparecfg_struct_para_init(&comparecfg_para);

comparecfg_para.compare_value = 192;

hrtimer_slavetimer_waveform_compare_config(HRTIMER0, HRTIMER_SLAVE_TIMER0,
HRTIMER_COMPARE0, &comparecfg_para);

```

## hrtimer\_channel\_outputcfg\_struct\_para\_init

The description of hrtimer\_channel\_outputcfg\_struct\_para\_init is shown as below:

**Table 3-977. Function hrtimer\_channel\_outputcfg\_struct\_para\_init**

<b>Function name</b>	hrtimer_channel_outputcfg_struct_para_init
<b>Function prototype</b>	void hrtimer_channel_outputcfg_struct_para_init(hrtimer_channel_outputcfg_parameter_struct * channelcfg);
<b>Function descriptions</b>	initialize channel output configuration struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelcfg</b>	hrtimer_channel_outputcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-957. Structure hrtimer_channel_outputcfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize channel output configuration struct with a default value */
hrtimer_channel_outputcfg_parameter_struct outcfg_para;
hrtimer_channel_outputcfg_struct_para_init(&outcfg_para);
```

## hrtimer\_slavetimer\_waveform\_channel\_config

The description of hrtimer\_slavetimer\_waveform\_channel\_config is shown as below:

**Table 3-978. Function hrtimer\_slavetimer\_waveform\_channel\_config**

<b>Function name</b>	hrtimer_slavetimer_waveform_channel_config
<b>Function prototype</b>	void hrtimer_slavetimer_waveform_channel_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t channel, hrtimer_channel_outputcfg_parameter_struct * channelcfg);
<b>Function descriptions</b>	configure the channel of a Slave_TIMER work in waveform mode
<b>Precondition</b>	-
<b>The called functions</b>	channel_output_config
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<b>HRTIMERx(x=0)</b>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index



<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	
<b>channel</b>	HRTIMER timer channel index
<i>HRTIMER_STx_CHy(x=0..7,y=0,1)</i>	HRTIMER timer channel selection
<b>Input parameter{in}</b>	
<b>channelcfg</b>	hrtimer_channel_outputcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-957. Structure hrtimer_channel_outputcfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure the channel of a Slave_TIMER work in waveform mode */

hrtimer_channel_outputcfg_parameter_struct outcfg_para;
hrtimer_channel_outputcfg_struct_para_init(&outcfg_para);

outcfg_para.carrier_mode = HRTIMER_CHANNEL_CARRIER_DISABLED;
outcfg_para.deadtime_bunch = HRTIMER_CHANNEL_BUNCH_ENTRY_REGULAR;
outcfg_para.fault_state = HRTIMER_CHANNEL_FAULTSTATE_NONE;
outcfg_para.idle_bunch = HRTIMER_CHANNEL_BUNCH_IDLE_DISABLE;
outcfg_para.idle_state = HRTIMER_CHANNEL_IDLESTATE_INACTIVE;
outcfg_para.polarity = HRTIMER_CHANNEL_POLARITY_HIGH;
outcfg_para.reset_request = HRTIMER_CHANNEL_RESET_CMP1;
outcfg_para.set_request = HRTIMER_CHANNEL_SET_CMP0;

hrtimer_slavetimer_waveform_channel_config(HRTIMER0, HRTIMER_SLAVE_TIMER0,
HRTIMER_ST0_CH0, &outcfg_para);

```

### **hrtimer\_slavetimer\_waveform\_channel\_software\_request**

The description of hrtimer\_slavetimer\_waveform\_channel\_software\_request is shown as below:

**Table 3-979. Function hrtimer\_slavetimer\_waveform\_channel\_software\_request**

<b>Function name</b>	hrtimer_slavetimer_waveform_channel_software_request
<b>Function prototype</b>	void hrtimer_slavetimer_waveform_channel_software_request(uint32_t

	hrtimer_periph, uint32_t timer_id, uint32_t channel, uint32_t request)
<b>Function descriptions</b>	software generates channel "set request" or "reset request"
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	
<b>channel</b>	HRTIMER timer channel index
<i>HRTIMER_STx_CHy(x=0..7,y=0,1)</i>	HRTIMER timer channel selection
<b>Input parameter{in}</b>	
<b>request</b>	request type selection
<i>HRTIMER_CHANNEL_SOFTWARE_SET</i>	software event cannot generate request
<i>HRTIMER_CHANNEL_SOFTWARE_RESET</i>	software event can generate request
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* software generates channel "set request" or "reset request" */
```

```
hrtimer_slavetimer_waveform_channel_software_request (HRTIMER0,
HRTIMER_SLAVE_TIMER0, HRTIMER_ST0_CH0,
HRTIMER_CHANNEL_SOFTWARE_SET);
```

### **hrtimer\_slavetimer\_waveform\_channel\_output\_level\_get**

The description of hrtimer\_slavetimer\_waveform\_channel\_output\_level\_get is shown as below:

**Table 3-980. Function hrtimer\_slavetimer\_waveform\_channel\_output\_level\_get**

<b>Function name</b>	hrtimer_slavetimer_waveform_channel_output_level_get
<b>Function prototype</b>	uint32_t hrtimer_slavetimer_waveform_channel_output_level_get(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t channel);
<b>Function descriptions</b>	get Slave_TIMER channel output level
<b>Precondition</b>	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	
<b>channel</b>	HRTIMER timer channel index
<i>HRTIMER_STx_CHy(x=0..7,y=0,1)</i>	HRTIMER timer channel selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	channel output level

Example:

```
/* get Slave_TIMER channel output level */
```

```
uint32_t output_level;
```

```
output_level = hrtimer_slavetimer_waveform_channel_output_level_get(uint32_t  
hrtimer_periph, uint32_t timer_id, uint32_t channel)
```

### **hrtimer\_slavetimer\_waveform\_channel\_state\_get**

The description of hrtimer\_slavetimer\_waveform\_channel\_state\_get is shown as below:

**Table 3-981. Function hrtimer\_slavetimer\_waveform\_channel\_state\_get**

<b>Function name</b>	hrtimer_slavetimer_waveform_channel_state_get
<b>Function prototype</b>	uint32_t hrtimer_slavetimer_waveform_channel_state_get(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t channel)
<b>Function descriptions</b>	get Slave_TIMER channel run state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	

<b>channel</b>	HRTIMER timer channel index
<i>HRTIMER_STx_CHy(x=0..7,y=0,1)</i>	HRTIMER timer channel selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	HRTIMER_CHANNEL_STATE_IDLE or HRTIMER_CHANNEL_STATE_RUN or HRTIMER_CHANNEL_STATE_FAULT

Example:

```
/* get Slave_TIMER channel run state */
```

```
uint32_t output_state;
```

```
output_state = hrtimer_slavetimer_waveform_channel_state_get (HRTIMER0,  
HRTIMER_SLAVE_TIMER0, HRTIMER_ST0_CH0);
```

### **hrtimer\_deadtimercfg\_struct\_para\_init**

The description of hrtimer\_deadtimercfg\_struct\_para\_init is shown as below:

**Table 3-982. Function hrtimer\_channel\_outputcfg\_struct\_para\_init**

<b>Function name</b>	hrtimer_deadtimercfg_struct_para_init
<b>Function prototype</b>	void hrtimer_deadtimercfg_struct_para_init(hrtimer_deadtimercfg_parameter_struct * dtcfg);
<b>Function descriptions</b>	initialize dead time configuration struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dtcfg</b>	hrtimer_deadtimercfg_parameter_struct, the structure members can refer to <a href="#">Table 3-949. Structure hrtimer_deadtimercfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize dead time configuration struct with a default value */
```

```
hrtimer_deadtimercfg_parameter_struct deadtimercfg_para;
```

```
hrtimer_deadtimercfg_struct_para_init(&deadtimercfg_para);
```

## hrtimer\_slavetimer\_deadtime\_config

The description of hrtimer\_slavetimer\_deadtime\_config is shown as below:

**Table 3-983. Function hrtimer\_slavetimer\_waveform\_channel\_software\_request**

<b>Function name</b>	hrtimer_slavetimer_deadtime_config
<b>Function prototype</b>	void hrtimer_slavetimer_deadtime_config(uint32_t hrtimer_periph, uint32_t timer_id, hrtimer_deadtimecfg_parameter_struct* dtcfg)
<b>Function descriptions</b>	configure the dead time for Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	
<b>dtcfg</b>	hrtimer_deadtimecfg_parameter_struct, the structure members can refer to <a href="#">Table 3-949. Structure hrtimer_deadtimecfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure the dead time for Slave_TIMER */

hrtimer_deadtimecfg_parameter_struct deadtimecfg_para;

hrtimer_deadtimecfg_struct_para_init(&deadtimecfg_para);

deadtimecfg_para.fallingsign_protect =
HRTIMER_DEADTIME_FALLINGSIGN_PROTECT_DISABLE;

deadtimecfg_para.falling_protect = HRTIMER_DEADTIME_FALLING_PROTECT_DISABLE;

deadtimecfg_para.falling_sign = HRTIMER_DEADTIME_FALLINGSIGN_POSITIVE;

deadtimecfg_para.falling_value = 0x0040;

deadtimecfg_para.prescaler = HRTIMER_DEADTIME_PRESCALER_MUL8;

deadtimecfg_para.risingsign_protect =
HRTIMER_DEADTIME_RISINGSIGN_PROTECT_DISABLE;

deadtimecfg_para.rising_protect = HRTIMER_DEADTIME_RISING_PROTECT_DISABLE;

```

```
deadtmeconfig_para.rising_sign = HRTIMER_DEADTIME_RISINGSIGN_POSITIVE;
```

```
deadtmeconfig_para.rising_value = 0x0040;
```

```
hrtimer_slavetimer_deadtme_config(HRTIMER0, HRTIMER_SLAVE_TIMER0,
&deadtmeconfig_para);
```

### hrtimer\_carriersignalcfg\_struct\_para\_init

The description of hrtimer\_carriersignalcfg\_struct\_para\_init is shown as below:

**Table 3-984. Function hrtimer\_channel\_outputcfg\_struct\_para\_init**

Function name	hrtimer_carriersignalcfg_struct_para_init
Function prototype	void hrtimer_carriersignalcfg_struct_para_init(hrtimer_carriersignalcfg_parameter_struct* carriercfg);
Function descriptions	initialize carrier signal configuration struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
carriercfg	hrtimer_carriersignalcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-950. Structure hrtimer_carriersignalcfg_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize carrier signal configuration struct with a default value */
```

```
hrtimer_carriersignalcfg_parameter_struct carriercfg;
```

```
hrtimer_carriersignalcfg_struct_para_init(&carriercfg);
```

### hrtimer\_slavetimer\_carriersignal\_config

The description of hrtimer\_slavetimer\_carriersignal\_config is shown as below:

**Table 3-985. Function hrtimer\_slavetimer\_carriersignal\_config**

Function name	hrtimer_slavetimer_carriersignal_config
Function prototype	void hrtimer_slavetimer_carriersignal_config(uint32_t hrtimer_periph, uint32_t timer_id, hrtimer_carriersignalcfg_parameter_struct* carriercfg);
Function descriptions	configure the carrier signal mode for Slave_TIMER
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral

<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	
<b>carriercfg</b>	hrtimer_carriersignalcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-950. Structure hrtimer_carriersignalcfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the carrier signal mode for Slave_TIMER */
hrtimer_carriersignalcfg_parameter_struct carriercfg;
hrtimer_carriersignalcfg_struct_para_init(&carriercfg);
hrtimer_slavetimer_carriersignal_config(HRTIMER0, HRTIMER_SLAVE_TIMER0,
&carriercfg);
```

### hrtimer\_output\_channel\_enable

The description of hrtimer\_output\_channel\_enable is shown as below:

**Table 3-986. Function hrtimer\_output\_channel\_enable**

<b>Function name</b>	hrtimer_output_channel_enable
<b>Function prototype</b>	void hrtimer_output_channel_enable(uint32_t hrtimer_periph, uint32_t chid);
<b>Function descriptions</b>	enable a output channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>ch_id</b>	HRTIMER timer channel index
<i>HRTIMER_STx_CHy(x=0..7;y=0,1)</i>	timer channel selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable a output channel */
```

```
void hrtimer_output_channel_enable(HRTIMER0, HRTIMER_ST0_CH0);
```

### hrtimer\_output\_channel\_disable

The description of hrtimer\_output\_channel\_disable is shown as below:

**Table 3-987. Function hrtimer\_output\_channel\_disable**

<b>Function name</b>	hrtimer_output_channel_disable
<b>Function prototype</b>	void hrtimer_output_channel_disable(uint32_t hrtimer_periph, uint32_t chid);
<b>Function descriptions</b>	disable a output channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>ch_id</b>	HRTIMER timer channel index
<i>HRTIMER_STx_CHy(x=0..7;y=0,1)</i>	timer channel selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable a output channel */
```

```
void hrtimer_output_channel_disable(HRTIMER0, HRTIMER_ST0_CH0);
```

### hrtimer\_slavetimer\_compare\_value\_config

The description of hrtimer\_slavetimer\_compare\_value\_config is shown as below:

**Table 3-988. Function hrtimer\_slavetimer\_waveform\_compare\_config**

<b>Function name</b>	hrtimer_slavetimer_compare_value_config
<b>Function prototype</b>	void hrtimer_slavetimer_compare_value_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t comparex, uint32_t cmpvalue);
<b>Function descriptions</b>	configure the compare value in Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection



Input parameter{in}	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
Input parameter{in}	
<b>comparex</b>	compare unit selection
<i>HRTIMER_COMPAREy(y=0..3)</i>	the compare unit y(y=0..3)
Input parameter{in}	
<b>cmpvalue</b>	the compare value from 3 tHRTIMER_CK clock to 0xFFFF - (1 tHRTIMER_CK)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the compare value in Slave_TIMER */
```

```
hrtimer_slavetimer_compare_value_config(HRTIMER0, HRTIMER_SLAVE_TIMER0,
HRTIMER_COMPARE0, 3);
```

### hrtimer\_slavetimer\_compare\_value\_get

The description of hrtimer\_slavetimer\_compare\_value\_get is shown as below:

**Table 3-989. Function hrtimer\_slavetimer\_compare\_value\_get**

<b>Function name</b>	hrtimer_slavetimer_compare_value_get
<b>Function prototype</b>	uint32_t hrtimer_slavetimer_compare_value_get(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t comparex)
<b>Function descriptions</b>	get the compare value in Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
Input parameter{in}	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
Input parameter{in}	
<b>comparex</b>	compare unit selection
<i>HRTIMER_COMPAREy(y=0..3)</i>	the compare unit y(y=0..3)

Output parameter{out}	
-	-
Return value	
uint32_t	the compare value

Example:

```
/* get the compare value in Slave_TIMER */
```

```
uint32_t cmpvalue;
```

```
cmpvalue = hrtimer_slavetimer_compare_value_get (HRTIMER0,
HRTIMER_SLAVE_TIMER0, HRTIMER_COMPARE0);
```

### hrtimer\_mastertimer\_compare\_value\_config

The description of hrtimer\_mastertimer\_compare\_value\_config is shown as below:

**Table 3-990. Function hrtimer\_mastertimer\_compare\_value\_config**

Function name	hrtimer_mastertimer_compare_value_config
Function prototype	void hrtimer_mastertimer_compare_value_config(uint32_t hrtimer_periph, uint32_t comparex, uint32_t cmpvalue);
Function descriptions	configure the compare value in Master_TIMER
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
comparex	compare unit selection
HRTIMER_COMPAREy (y=0..3)	the compare unit y(y=0..3)
Input parameter{in}	
cmpvalue	the compare value from 3 tHRTIMER_CK clock to 0xFFFF - (1 tHRTIMER_CK)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the compare value in Master_TIMER */
```

```
hrtimer_mastertimer_compare_value_config(HRTIMER0, HRTIMER_COMPARE0, 3);
```

## hrtimer\_mastertimer\_compare\_value\_get

The description of hrtimer\_mastertimer\_compare\_value\_get is shown as below:

**Table 3-991. Function hrtimer\_slavetimer\_compare\_value\_get**

<b>Function name</b>	hrtimer_mastertimer_compare_value_get
<b>Function prototype</b>	uint32_t hrtimer_mastertimer_compare_value_get(uint32_t hrtimer_periph, uint32_t comparex);
<b>Function descriptions</b>	get the compare value in Master_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>comparex</b>	compare unit selection
<i>HRTIMER_COMPAREy</i> (y=0..3)	the compare unit y(y=0..3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the compare value

Example:

```
/* get the compare value in Master_TIMER */
```

```
uint32_t cmpvalue;
```

```
cmpvalue = hrtimer_mastertimer_compare_value_get(HRTIMER0, HRTIMER_COMPARE0)
```

## hrtimer\_timers\_counter\_value\_config

The description of hrtimer\_timers\_counter\_value\_config is shown as below:

**Table 3-992. Function hrtimer\_timers\_counter\_value\_config**

<b>Function name</b>	hrtimer_timers_counter_value_config
<b>Function prototype</b>	void hrtimer_timers_counter_value_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t cntvalue);
<b>Function descriptions</b>	configure the counter value in Master_TIMER and Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	

<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_MASTER_TIMER</i>	the counter of Master_TIMER
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	
<b>cntvalue</b>	the value ranges from 0 to 0xffff
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the counter value in Master_TIMER and Slave_TIMER */
```

```
hrtimer_timers_counter_value_config(HRTIMER0, HRTIMER_MASTER_TIMER, 100);
```

### **hrtimer\_timers\_counter\_value\_get**

The description of hrtimer\_timers\_counter\_value\_get is shown as below:

**Table 3-993. Function hrtimer\_timers\_counter\_value\_get**

<b>Function name</b>	hrtimer_timers_counter_value_get
<b>Function prototype</b>	uint32_t hrtimer_timers_counter_value_get(uint32_t hrtimer_periph, uint32_t timer_id);
<b>Function descriptions</b>	get the counter value in Master_TIMER and Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_MASTER_TIMER</i>	the counter of Master_TIMER
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the counter value

Example:

```
/* get the counter value in Master_TIMER and Slave_TIMER */
```

```
uint32_t value;
```

```
value = hrtimer_timers_counter_value_get(HRTIMER0, HRTIMER_MASTER_TIMER);
```

### hrtimer\_timers\_autoreload\_value\_config

The description of hrtimer\_timers\_autoreload\_value\_config is shown as below:

**Table 3-994. Function hrtimer\_timers\_autoreload\_value\_config**

Function name	hrtimer_timers_autoreload_value_config
Function prototype	void hrtimer_timers_autoreload_value_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t carlvalue);
Function descriptions	configure the counter auto reload value in Master_TIMER and Slave_TIMER
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
timer_id	HRTIMER peripheral
HRTIMER_MASTER_TIMER	the counter of Master_TIMER
HRTIMER_SLAVE_TIMERx(x=0..7)	the counter of Slave_TIMERx(x=0..7)
Input parameter{in}	
carlvalue	the value ranges from 0 to 0xffff
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the counter auto reload value in Master_TIMER and Slave_TIMER */
```

```
hrtimer_timers_autoreload_value_config(HRTIMER0, HRTIMER_MASTER_TIMER, 100);
```

### hrtimer\_timers\_autoreload\_value\_get

The description of hrtimer\_timers\_autoreload\_value\_get is shown as below:

**Table 3-995. Function hrtimer\_timers\_autoreload\_value\_get**

Function name	hrtimer_timers_autoreload_value_get
Function prototype	uint32_t hrtimer_timers_autoreload_value_get(uint32_t hrtimer_periph, uint32_t timer_id)
Function descriptions	get the counter auto reload value in Master_TIMER and Slave_TIMER

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_MASTER_TIMER</i>	the counter of Master_TIMER
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the counter value

Example:

```
/* get the counter auto reload value in Master_TIMER and Slave_TIMER */
```

```
uint32_t value;
```

```
value = hrtimer_timers_autoreload_value_get(HRTIMER0, HRTIMER_MASTER_TIMER);
```

### **hrtimer\_timers\_repetition\_value\_config**

The description of hrtimer\_timers\_repetition\_value\_config is shown as below:

**Table 3-996. Function hrtimer\_timers\_repetition\_value\_config**

<b>Function name</b>	hrtimer_timers_repetition_value_config
<b>Function prototype</b>	void hrtimer_timers_repetition_value_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t replvalue);
<b>Function descriptions</b>	configure the counter repetition value in Master_TIMER and Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_MASTER_TIMER</i>	the counter of Master_TIMER
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	

<b>replvalue</b>	the value ranges from 0 to 0xffff
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the counter repetition value in Master_TIMER and Slave_TIMER */
hrtimer_timers_repetition_value_config(HRTIMER0, HRTIMER_MASTER_TIMER, 100);
```

### **hrtimer\_timers\_repetition\_value\_get**

The description of hrtimer\_timers\_repetition\_value\_get is shown as below:

**Table 3-997. Function hrtimer\_timers\_repetition\_value\_get**

<b>Function name</b>	hrtimer_timers_repetition_value_get
<b>Function prototype</b>	uint32_t hrtimer_timers_repetition_value_get(uint32_t hrtimer_periph, uint32_t timer_id);
<b>Function descriptions</b>	get the counter repetition value in Master_TIMER and Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_MASTER_TIMER</i>	the counter of Master_TIMER
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the counter value

Example:

```
/* get the counter repetition value in Master_TIMER and Slave_TIMER */
uint32_t value;
value = hrtimer_timers_repetition_value_get (HRTIMER0, HRTIMER_MASTER_TIMER);
```

### **hrtimer\_exeefilter\_struct\_para\_init**

The description of hrtimer\_exeefilter\_struct\_para\_init is shown as below:

Table 3-998. Function `hrtimer_exevfilter_struct_para_init`

Function name	<code>hrtimer_exevfilter_struct_para_init</code>
Function prototype	<code>void hrtimer_exevfilter_struct_para_init(hrtimer_exevfilter_parameter_struct * exevfilter);</code>
Function descriptions	initialize external event filtering for Slave_TIMER configuration struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
<b>exevfilter</b>	<code>hrtimer_exevfilter_parameter_struct * exevfilter</code> , the structure members can refer to <a href="#">Table 3-948. Structure <code>hrtimer_exevfilter_parameter_struct</code></a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize external event filtering for Slave_TIMER configuration struct with a default value */
```

```
hrtimer_exevfilter_parameter_struct exevfilter;
```

```
hrtimer_exevfilter_struct_para_init(&exevfilter);
```

### **`hrtimer_slavetimer_exeevent_filtering_config`**

The description of `hrtimer_slavetimer_exeevent_filtering_config` is shown as below:

Table 3-999. Function `hrtimer_slavetimer_exeevent_filtering_config`

Function name	<code>hrtimer_slavetimer_exeevent_filtering_config</code>
Function prototype	<code>void hrtimer_slavetimer_exeevent_filtering_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t event_id, hrtimer_exevfilter_parameter_struct *exevfilter);</code>
Function descriptions	configure the external event filtering for Slave_TIMER (blanking, windowing)
Precondition	-
The called functions	-
Input parameter{in}	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
Input parameter{in}	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
Input parameter{in}	
<b>event_id</b>	HRTIMER event index
<i>HRTIMER_EXEVENT_</i>	the counter of Slave_TIMERx <i>undefined event channel</i>



NONE	
HRTIMER_EXEVENT_ y(y=0..9)	extern event y(y=0..9)
Input parameter{in}	
exevfilter	hrtimer_exevfilter_parameter_struct * exevfilter, the structure members can refer to <a href="#">Table 3-948. Structure hrtimer_exevfilter_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure the external event filtering for Slave_TIMER (blanking, windowing) */

hrtimer_exevfilter_parameter_struct exevfilter;

hrtimer_exevfilter_struct_para_init(&exevfilter);

exevfilter.filter_mode = HRTIMER_EXEVFILTER_BLANKINGCMP1;

exevfilter.memorized = HRTIMER_EXEVMEMORIZED_DISABLE;

hrtimer_slavetimer_exeevent_filtering_config(HRTIMER0, HRTIMER_SLAVE_TIMER0,
HRTIMER_EXEVENT_5, &exevfilter);

```

### hrtimer\_exeeventcfg\_struct\_para\_init

The description of hrtimer\_exeeventcfg\_struct\_para\_init is shown as below:

**Table 3-1000. Function hrtimer\_exeeventcfg\_struct\_para\_init**

Function name	hrtimer_exeeventcfg_struct_para_init
Function prototype	void hrtimer_exeeventcfg_struct_para_init(hrtimer_exeeventcfg_parameter_struct * exevcfg);
Function descriptions	initialize external event configuration struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
exevcfg	hrtimer_exeeventcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-953. Structure hrtimer_exeeventcfg_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize external event configuration struct with a default value */
```

```
hrtimer_exeventcfg_parameter_struct exevcfg;
```

```
hrtimer_exeventcfg_struct_para_init(&exevcfg);
```

### hrtimer\_exevent\_config

The description of hrtimer\_exevent\_config is shown as below:

**Table 3-1001. Function hrtimer\_exevent\_config**

Function name	hrtimer_exevent_config
Function prototype	void hrtimer_exevent_config(uint32_t hrtimer_periph, uint32_t event_id, hrtimer_exeventcfg_parameter_struct* exevcfg);
Function descriptions	configure the an external event
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
event_id	HRTIMER event index
HRTIMER_EXEVENT_NONE	the counter of Slave_TIMERx <i>undefined event channel</i>
HRTIMER_EXEVENT_y(y=0..9)	extern event y(y=0..9)
Input parameter{in}	
exevcfg	hrtimer_exeventcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-953. Structure hrtimer_exeventcfg_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the an external event */
```

```
hrtimer_exeventcfg_parameter_struct exevcfg;
```

```
hrtimer_exeventcfg_struct_para_init(&exevcfg);
```

```
exevcfg.digital_filter = 0x5;
```

```
exevcfg.edge = HRTIMER_EXEV_EDGE_RISING;
```

```
exevcfg.polarity = HRTIMER_EXEV_EDGE_LEVEL;
```

```
exevcfg.source = HRTIMER_EXEV_SRC0;
```

```
hrtimer_exevent_config(HRTIMER0, HRTIMER_EXEVENT_5, &exevcfg);
```

## hrtimer\_exevent\_prescaler

The description of hrtimer\_exevent\_prescaler is shown as below:

**Table 3-1002. Function hrtimer\_exevent\_prescaler**

<b>Function name</b>	hrtimer_exevent_prescaler
<b>Function prototype</b>	void hrtimer_exevent_prescaler(uint32_t hrtimer_periph, uint32_t prescaler);
<b>Function descriptions</b>	configure external event digital filter clock division
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>prescaler</b>	clock division value
<i>HRTIMER_EXEV_PRE SCALER_DIVx(x=1,2,4, 8)</i>	$f_{HRTIMER\_EXEV\_FCK} = f_{HRTIMER\_CK}/x$ (x=1,2,4,8)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure external event digital filter clock division */
```

```
hrtimer_exevent_prescaler(HRTIMER0, HRTIMER_EXEV_PRESCALER_DIV1);
```

## hrtimer\_exeventx\_counter\_struct\_para\_init

The description of hrtimer\_exeventx\_counter\_struct\_para\_init is shown as below:

**Table 3-1003. Function hrtimer\_exevent\_config**

<b>Function name</b>	hrtimer_exeventx_counter_struct_para_init
<b>Function prototype</b>	void hrtimer_exeventx_counter_struct_para_init(hrtimer_exeventcnt_parameter_ struct *exevcnt);
<b>Function descriptions</b>	initialize external event configuration struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection

Input parameter{in}	
<b>exevcnt</b>	external event counter struct, the structure members can refer to <a href="#">Table 3-954. Structure hrtimer exeventcnt parameter struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize the an external event struct with a default value */
```

```
hrtimer_exeventcnt_parameter_struct exevcnt;
```

```
hrtimer_exeventx_counter_struct_para_init(&exevcfg);
```

### hrtimer\_exeventx\_counter\_config

The description of hrtimer\_exeventx\_counter\_config is shown as below:

**Table 3-1004. Function hrtimer\_exeventx\_counter\_config**

Function name	hrtimer_exeventx_counter_config
Function prototype	void hrtimer_exeventx_counter_config(uint32_t hrtimer_periph, uint32_t timer_id, hrtimer_exeventcnt_parameter_struct *exevcnt);
Function descriptions	configure the external event X counter
Precondition	-
The called functions	-
Input parameter{in}	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
Input parameter{in}	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
Input parameter{in}	
<b>exevcnt</b>	hrtimer_exeventcnt_parameter_struct, the structure members can refer to <a href="#">Table 3-954. Structure hrtimer exeventcnt parameter struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the external event X counter */
```

```
hrtimer_exeventcnt_parameter_struct exevcnt;
```

```

hrtimer_exeventx_counter_struct_para_init (&exevcnt);

exevcnt.exevcnt = 0x5;

exevcnt.reset_mode = HRTIMER_EXEVX_RESET_MODE0;

exevcnt.event_source = HRTIMER_EXEVX_SOURCE_EXEV0;

hrtimer_exeventx_counter_config(HRTIMER0, HRTIMER_SLAVE_TIMER0, &exevcnt);

```

### hrtimer\_software\_reset\_exeventx\_counter

The description of hrtimer\_software\_reset\_exeventx\_counter is shown as below:

**Table 3-1005. Function hrtimer\_software\_reset\_exeventx\_counter**

<b>Function name</b>	hrtimer_software_reset_exeventx_counter
<b>Function prototype</b>	void hrtimer_software_reset_exeventx_counter(uint32_t hrtimer_periph, uint32_t timer_id);
<b>Function descriptions</b>	reset external event X counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* reset external event X counter */

hrtimer_software_reset_exeventx_counter(HRTIMER0, HRTIMER_SLAVE_TIMER0);

```

### hrtimer\_exeventx\_counter\_enable

The description of hrtimer\_exeventx\_counter\_enable is shown as below:

**Table 3-1006. Function hrtimer\_exeventx\_counter\_enable**

<b>Function name</b>	hrtimer_exeventx_counter_enable
<b>Function prototype</b>	void hrtimer_exeventx_counter_enable(uint32_t hrtimer_periph, uint32_t timer_id);
<b>Function descriptions</b>	enable external event X counter

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable external event X counter */
```

```
hrtimer_exeventx_counter_enable(HRTIMER0, HRTIMER_SLAVE_TIMER0);
```

### **hrtimer\_exeventx\_counter\_disable**

The description of hrtimer\_exeventx\_counter\_disable is shown as below:

**Table 3-1007. Function hrtimer\_exeventx\_counter\_disable**

<b>Function name</b>	hrtimer_exeventx_counter_disable
<b>Function prototype</b>	void hrtimer_exeventx_counter_disable(uint32_t hrtimer_periph, uint32_t timer_id);
<b>Function descriptions</b>	disable external event X counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER timer index
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable external event X counter */
```

```
hrtimer_exeventx_counter_disable(HRTIMER0, HRTIMER_SLAVE_TIMER0);
```

## hrtimer\_exeventx\_counter\_read

The description of hrtimer\_exeventx\_counter\_read is shown as below:

**Table 3-1008. Function hrtimer\_exeventx\_counter\_read**

<b>Function name</b>	hrtimer_exeventx_counter_read
<b>Function prototype</b>	uint32_t hrtimer_exeventx_counter_read(uint32_t hrtimer_periph, uint32_t timer_id)
<b>Function descriptions</b>	read external event X counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the external event X counter value

Example:

```
/* get the external event X counter value */
```

```
uint32_t value;
```

```
value = hrtimer_exeventx_counter_read (HRTIMER0, HRTIMER_SLAVE_TIMER0);
```

## hrtimer\_synccfg\_struct\_para\_init

The description of hrtimer\_synccfg\_struct\_para\_init is shown as below:

**Table 3-1009. Function hrtimer\_synccfg\_struct\_para\_init**

<b>Function name</b>	hrtimer_synccfg_struct_para_init
<b>Function prototype</b>	void hrtimer_synccfg_struct_para_init(hrtimer_synccfg_parameter_struct* synccfg);
<b>Function descriptions</b>	initialize synchronization configuration struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>synccfg</b>	hrtimer_synccfg_parameter_struct, the structure members can refer to <a href="#">Table 3-951. Structure hrtimer_synccfg_parameter_struct</a>

Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize synchronization configuration struct with a default value */

hrtimer_synccfg_parameter_struct synccfg;

hrtimer_synccfg_struct_para_init (&synccfg);

```

### hrtimer\_synchronization\_config

The description of hrtimer\_synchronization\_config is shown as below:

**Table 3-1010. Function hrtimer\_synchronization\_config**

Function name	hrtimer_synchronization_config
Function prototype	void hrtimer_synchronization_config(uint32_t hrtimer_periph, hrtimer_synccfg_parameter_struct* synccfg);
Function descriptions	configure the synchronization input/output of the HRTIMER
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
synccfg	hrtimer_synccfg_parameter_struct, the structure members can refer to <a href="#">Table 3-951. Structure hrtimer_synccfg_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure the synchronization input/output of the HRTIMER */

hrtimer_synccfg_parameter_struct synccfg;

hrtimer_synccfg_struct_para_init (&synccfg);

synccfg.input_source = HRTIMER_SYNCINPUTSOURCE_EXTERNAL;

hrtimer_synchronization_config(HRTIMER0, &synccfg);

```



## hrtimer\_double\_channel\_struct\_para\_init

The description of hrtimer\_double\_channel\_struct\_para\_init is shown as below:

**Table 3-1011. Function hrtimer\_double\_channel\_struct\_para\_init**

<b>Function name</b>	hrtimer_double_channel_struct_para_init
<b>Function prototype</b>	void hrtimer_double_channel_struct_para_init(void) hrtimer_double_channel_struct_para_init(hrtimer_double_trigger_parameter_struct *doubletrigger);
<b>Function descriptions</b>	initialize hrtimer_double_trigger_parameter_struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>doubletrigger</b>	hrtimer_double_trigger_parameter_struct, the structure members can refer to <a href="#">Table 3-959. Structure</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize double trigger struct with a default value */
```

```
hrtimer_double_trigger_parameter_struct doubletrigger;
```

```
hrtimer_double_channel_struct_para_init(&doubletrigger);
```

## hrtimer\_double\_trigger\_config

The description of hrtimer\_double\_trigger\_config is shown as below:

**Table 3-1012. Function hrtimer\_double\_trigger\_config**

<b>Function name</b>	hrtimer_double_trigger_config
<b>Function prototype</b>	void hrtimer_double_trigger_config(uint32_t hrtimer_periph, uint32_t timer_id, hrtimer_double_trigger_parameter_struct *doubletrigger);
<b>Function descriptions</b>	configure Slave_TIMER in double channel trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<b>HRTIMERx(x=0)</b>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>HRTIMER_SLAVE_TIMERx(x=0..7)</b>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	

<b>doubletrigger</b>	hrtimer_double_trigger_parameter_struct, the structure members can refer to <a href="#">Table 3-959. Structure hrtimer_double_trigger_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure Slave_TIMER in double channel trigger */
hrtimer_double_trigger_parameter_struct doubletrigger;
hrtimer_double_trigger_config (HRTIMER0, HRTIMER_SLAVE_TIMER0, & doubletrigger);
```

### hrtimer\_roll\_over\_struct\_para\_init

The description of hrtimer\_roll\_over\_struct\_para\_init is shown as below:

**Table 3-1013. Function hrtimer\_roll\_over\_struct\_para\_init**

<b>Function name</b>	hrtimer_roll_over_struct_para_init
<b>Function prototype</b>	void hrtimer_roll_over_struct_para_init(hrtimer_roll_over_parameter_struct *rollover);
<b>Function descriptions</b>	initialize roll over struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rollover</b>	hrtimer_roll_over_parameter_struct, the structure members can refer to <a href="#">Table 3-958. Structure hrtimer_roll_over_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize roll over struct with a default value */
hrtimer_roll_over_parameter_struct rollover;
hrtimer_roll_over_struct_para_init (&rollover);
```

### hrtimer\_roll\_over\_mode\_config

The description of hrtimer\_roll\_over\_mode\_config is shown as below:

**Table 3-1014. Function hrtimer\_roll\_over\_mode\_config**

<b>Function name</b>	hrtimer_roll_over_mode_config
----------------------	-------------------------------

<b>Function prototype</b>	void hrtimer_roll_over_mode_config(uint32_t hrtimer_periph, uint32_t timer_id, hrtimer_roll_over_parameter_struct *rollover);
<b>Function descriptions</b>	configure Slave_TIMER in roll over mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<b>HRTIMERx(x=0)</b>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>HRTIMER_SLAVE_TIMERx(x=0..7)</b>	the counter of Slave_TIMERx(x=0..7)
<b>Input parameter{in}</b>	
<b>rollover</b>	hrtimer_roll_over_parameter_struct, the structure members can refer to <a href="#">Table 3-958. Structure hrtimer_roll_over_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure Slave_TIMER in roll over mode */

hrtimer_roll_over_parameter_struct rollover;

hrtimer_roll_over_struct_para_init (&rollover);

rollover.roll_over_mode = HRTIMER_ROLLOVER_MODE1;

rollover.output_roll_over_mode = HRTIMER_OUTPUT_ROLLOVER_MODE0;

rollover.adc_roll_over_mode = HRTIMER_ADC_ROLLOVER_MODE0;

rollover.bunch_mode_roll_over_mode = HRTIMER_BUNCH_MODE_ROLLOVER_MODE0;

rollover.fault_event_roll_over_mode = HRTIMER_FAULT_EVENTROLLOVER_MODE0;

hrtimer_roll_over_mode_config (HRTIMER0 , HRTIMER_SLAVE_TIMER0, &rollover);

```

### hrtimer\_faultcfg\_struct\_para\_init

The description of hrtimer\_faultcfg\_struct\_para\_init is shown as below:

**Table 3-1015. Function hrtimer\_faultcfg\_struct\_para\_init**

<b>Function name</b>	hrtimer_faultcfg_struct_para_init
<b>Function prototype</b>	void hrtimer_faultcfg_struct_para_init(hrtimer_faultcfg_parameter_struct * faultcfg);

<b>Function descriptions</b>	initialize fault input configuration struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>faultcfg</b>	hrtimer_faultcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-955. Structure hrtimer_faultcfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize fault input configuration struct with a default value */
```

```
hrtimer_faultcfg_parameter_struct faultcfg;
```

```
hrtimer_synccfg_struct_para_init (&faultcfg);
```

### hrtimer\_fault\_config

The description of hrtimer\_fault\_config is shown as below:

**Table 3-1016. Function hrtimer\_fault\_config**

<b>Function name</b>	hrtimer_fault_config
<b>Function prototype</b>	void hrtimer_fault_config(uint32_t hrtimer_periph, uint32_t fault_id, hrtimer_faultcfg_parameter_struct* faultcfg);
<b>Function descriptions</b>	configure the fault input
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>fault_id</b>	HRTIMER fault index
<i>HRTIMER_FAULT_y(y=0..7)</i>	HRTIMER fault selection
<b>Input parameter{in}</b>	
<b>faultcfg</b>	hrtimer_faultcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-955. Structure hrtimer_faultcfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure the fault input */

hrtimer_faultcfg_parameter_struct faultcfg;

hrtimer_synccfg_struct_para_init (&faultcfg);

faultcfg_para.control = HRTIMER_FAULT_CHANNEL_ENABLE;

faultcfg_para.filter = 0x0;

faultcfg_para.polarity = HRTIMER_FAULT_POLARITY_HIGH;

faultcfg_para.protect = HRTIMER_FAULT_PROTECT_ENABLE;

faultcfg_para.source = HRTIMER_FAULT_SOURCE_PIN;

faultcfg_para.blanksource = HRTIMER_FAULT_COMP_COMP;

faultcfg_para.counter = 3;

faultcfg_para.resetmode = HRTIMER_FAULT_COUNTER_RESET_CONDITIONAL;

hrtimer_fault_config(HRTIMER0, HRTIMER_FAULT_0, &faultcfg);

```

### hrtimer\_fault\_prescaler\_config

The description of hrtimer\_fault\_prescaler\_config is shown as below:

**Table 3-1017. Function hrtimer\_fault\_prescaler\_config**

Function name	hrtimer_fault_prescaler_config
Function prototype	void hrtimer_fault_prescaler_config(uint32_t hrtimer_periph, uint32_t prescaler);
Function descriptions	configure the fault input digital filter clock division
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
prescaler	clock division value
HRTIMER_FAULT_PRESCALER_DIVy(y=1,2,4,8)	$f_{HRTIMER\_FLTCK} = f_{HRTIMER\_CK}/y$ (y=1,2,4,8)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure the fault input digital filter clock division */

```

```
void hrtimer_fault_prescaler_config(HRTIMER0, HRTIMER_FAULT_PRESCALER_DIV1);
```

## hrtimer\_fault\_input\_enable

The description of hrtimer\_fault\_input\_enable is shown as below:

**Table 3-1018. Function hrtimer\_fault\_input\_enable**

<b>Function name</b>	hrtimer_fault_input_enable
<b>Function prototype</b>	void hrtimer_fault_input_enable(uint32_t hrtimer_periph, uint32_t fault_id);
<b>Function descriptions</b>	fault input enable
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>fault_id</b>	HRTIMER fault index
<i>HRTIMER_FAULT_y(y=0..7)</i>	HRTIMER fault selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* fault input enable */
```

```
hrtimer_fault_input_enable(HRTIMER0, HRTIMER_FAULT_0);
```

## hrtimer\_fault\_input\_disable

The description of hrtimer\_fault\_input\_disable is shown as below:

**Table 3-1019. Function hrtimer\_fault\_input\_disable**

<b>Function name</b>	hrtimer_fault_input_disable
<b>Function prototype</b>	void hrtimer_fault_input_disable(uint32_t hrtimer_periph, uint32_t fault_id);
<b>Function descriptions</b>	fault input disable
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>fault_id</b>	HRTIMER fault index
<i>HRTIMER_FAULT_y(y=0..7)</i>	HRTIMER fault selection

=0..7)	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* fault input disable */
```

```
hrtimer_fault_input_disable(HRTIMER0, HRTIMER_FAULT_0);
```

### hrtimer\_fault\_counter\_reset

The description of hrtimer\_fault\_counter\_reset is shown as below:

**Table 3-1020. Function hrtimer\_fault\_counter\_reset**

<b>Function name</b>	hrtimer_fault_counter_reset
<b>Function prototype</b>	void hrtimer_fault_counter_reset(uint32_t hrtimer_periph, uint32_t fault_id);
<b>Function descriptions</b>	fault input couter reset
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
Input parameter{in}	
<b>fault_id</b>	HRTIMER fault index
<i>HRTIMER_FAULT_y(y=0..7)</i>	HRTIMER fault selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* fault input couter reset */
```

```
hrtimer_fault_counter_reset(HRTIMER0, HRTIMER_FAULT_0);
```

### hrtimer\_fault\_blank\_enable

The description of hrtimer\_fault\_blank\_enable is shown as below:

**Table 3-1021. Function hrtimer\_fault\_blank\_enable**

<b>Function name</b>	hrtimer_fault_blank_enable
<b>Function prototype</b>	void hrtimer_fault_blank_enable(uint32_t hrtimer_periph, uint32_t fault_id);

<b>Function descriptions</b>	fault input blank enable
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx</i> (x=0)	HRTIMER selection
<b>Input parameter{in}</b>	
<b>fault_id</b>	HRTIMER fault index
<i>HRTIMER_FAULT_y</i> (y=0..7)	HRTIMER fault selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* fault input blank enable*/
```

```
hrtimer_fault_blank_enable(HRTIMER0, HRTIMER_FAULT_0);
```

### hrtimer\_fault\_blank\_disable

The description of hrtimer\_fault\_blank\_disable is shown as below:

**Table 3-1022. Function hrtimer\_fault\_blank\_disable**

<b>Function name</b>	hrtimer_fault_blank_disable
<b>Function prototype</b>	void hrtimer_fault_blank_disable (uint32_t hrtimer_periph, uint32_t fault_id);
<b>Function descriptions</b>	fault input blank disable
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx</i> (x=0)	HRTIMER selection
<b>Input parameter{in}</b>	
<b>fault_id</b>	HRTIMER fault index
<i>HRTIMER_FAULT_y</i> (y=0..7)	HRTIMER fault selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* fault input blank disable*/
```



```
hrtimer_fault_blank_disable(HRTIMER0, HRTIMER_FAULT_0);
```

### hrtimer\_timers\_dma\_enable

The description of hrtimer\_timers\_dma\_enable is shown as below:

**Table 3-1023. Function hrtimer\_timers\_dma\_enable**

Function name	hrtimer_timers_dma_enable
Function prototype	void hrtimer_timers_dma_enable(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t dmareq);
Function descriptions	enable the Master_TIMER and Slave_TIMER DMA request
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
timer_id	master timer and slave timer index
HRTIMER_MASTER_TIMER	the master timer
HRTIMER_SLAVE_TIMERx	the slave timer selection(x=0..7)
Input parameter{in}	
dmareq	DMA request source
HRTIMER_MT_ST_DMA_CMPy(y=0..3)	compare y DMA request, for Master_TIMER and Slave_TIMER (y=0..3)
HRTIMER_MT_ST_DMA_A_REP	repetition DMA request, for Master_TIMER and Slave_TIMER
HRTIMER_MT_DMA_SYNCID	synchronization input DMA request, for Master_TIMER
HRTIMER_MT_ST_DMA_A_UPD	update DMA request, for Master_TIMER and Slave_TIMER
HRTIMER_ST_DMA_CAPy(y=0,1)	capture y DMA request, for Slave_TIMER(y=0,1)
HRTIMER_ST_DMA_CHyOA(y=0,1)	channel y output active DMA request, for Slave_TIMER(y=0,1)
HRTIMER_ST_DMA_CHyONA(y=0,1)	channel y output inactive DMA request, for Slave_TIMER(y=0,1)
HRTIMER_ST_DMA_CNTRST	counter reset DMA request, for Slave_TIMER
HRTIMER_ST_DMA_DELAYIDLE	delayed IDLE mode entry DMA request, for Slave_TIMER
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* enable the Master_TIMER and Slave_TIMER DMA request */
```

```
hrtimer_timers_dma_enable(HRTIMER0, HRTIMER_MASTER_TIMER,
HRTIMER_MT_ST_DMA_CMP0);
```

### hrtimer\_timers\_dma\_disable

The description of hrtimer\_timers\_dma\_disable is shown as below:

**Table 3-1024. Function hrtimer\_timers\_dma\_enable**

Function name	hrtimer_timers_dma_disable
Function prototype	void hrtimer_timers_dma_disable(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t dmareq);
Function descriptions	disable the Master_TIMER and Slave_TIMER DMA request
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
timer_id	master timer and slave timer index
HRTIMER_MASTER_TIMER	the master timer
HRTIMER_SLAVE_TIMERx	the slave timer selection(x=0..7)
Input parameter{in}	
dmareq	DMA request source
HRTIMER_MT_ST_DMA_CMPy(y=0..3)	compare y DMA request, for Master_TIMER and Slave_TIMER (y=0..3)
HRTIMER_MT_ST_DMA_A_REP	repetition DMA request, for Master_TIMER and Slave_TIMER
HRTIMER_MT_DMA_SYNCID	synchronization input DMA request, for Master_TIMER
HRTIMER_MT_ST_DMA_A_UPD	update DMA request, for Master_TIMER and Slave_TIMER
HRTIMER_ST_DMA_CAPTUREy(y=0,1)	capture y DMA request, for Slave_TIMER(y=0,1)
HRTIMER_ST_DMA_CHANNELyOA(y=0,1)	channel y output active DMA request, for Slave_TIMER(y=0,1)
HRTIMER_ST_DMA_CHANNELyIN(y=0,1)	channel y output inactive DMA request, for Slave_TIMER(y=0,1)

<i>HyONA(y=0,1)</i>	
<i>HRTIMER_ST_DMA_CNTRST</i>	counter reset DMA request, for Slave_TIMER
<i>HRTIMER_ST_DMA_DELAYIDLE</i>	delayed IDLE mode entry DMA request, for Slave_TIMER
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the Master_TIMER and Slave_TIMER DMA request */
```

```
hrtimer_timers_dma_disable(HRTIMER0, HRTIMER_MASTER_TIMER,
HRTIMER_MT_ST_DMA_CMP0);
```

### hrtimer\_dmamode\_config

The description of hrtimer\_dmamode\_config is shown as below:

**Table 3-1025. Function hrtimer\_dmamode\_config**

<b>Function name</b>	hrtimer_dmamode_config
<b>Function prototype</b>	void hrtimer_dmamode_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t regupdate);
<b>Function descriptions</b>	configure the DMA mode for Master_TIMER or Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_MASTER_TIMER</i>	the master timer
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0.. <b>7</b> )
<b>Input parameter{in}</b>	
<b>regupdate</b>	registers to be updated
<i>HRTIMER_DMAMODE_NONE</i>	No register is updated by DMA mode, for Master_TIMER and Slave_TIMER
<i>HRTIMER_DMAMODE_CTL0</i>	MTCTL0 or STxCTL0 register is updated by DMA mode, for Master_TIMER and Slave_TIMER
<i>HRTIMER_DMAMODE_INTC</i>	MT or STx register is updated by DMA mode, for Master_TIMER and Slave_TIMER

<i>HRTIMER_DMAMODE</i> <i>_DMAINTEN</i>	MTINTC or STxINTC register is updated by DMA mode, for Master_TIMER and Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_CNT</i>	MTCNT or STxCNT register is updated by DMA mode, for Master_TIMER and Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_CAR</i>	MTCAR or STxCAR register is updated by DMA mode, for Master_TIMER and Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_CREP</i>	MTCREP or STxCREP register is updated by DMA mode, for Master_TIMER and Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_CMPyV(y=0..3)</i>	MTCMPyV or STxCMPyV register is updated by DMA mode, for Master_TIMER and Slave_TIMER(y=0..3)
<i>HRTIMER_DMAMODE</i> <i>_DTCTL</i>	STxDTCCTL register is updated by DMA mode, only for Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_CHySET(y=0,1)</i>	STxCHySET register is updated by DMA mode, only for Slave_TIMER(y=0,1)
<i>HRTIMER_DMAMODE</i> <i>_CHyRST(y=0,1)</i>	STxCHyRST register is updated by DMA mode, only for Slave_TIMER(y=0,1)
<i>HRTIMER_DMAMODE</i> <i>_EXEVFCFGy(y=0,1)</i>	STxEXEVFCFGy register is updated by DMA mode, only for Slave_TIMER(y=0,1)
<i>HRTIMER_DMAMODE</i> <i>_CNTRST</i>	STxCNTRST register is updated by DMA mode, only for Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_CSCTL</i>	STxCSCCTL register is updated by DMA mode, only for Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_CHOCTL</i>	STxCHOCTL register is updated by DMA mode, only for Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_CTL1</i>	STxCTL1 register is updated by DMA mode, only for Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_EXEVFCFG2</i>	STxEXEVFCFG2 register is updated by DMA mode, only for Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_FLTCTL</i>	STxFLTCTL register is updated by DMA mode, only for Slave_TIMER
<i>HRTIMER_DMAMODE</i> <i>_ACTL</i>	STxACTL register is updated by DMA mode, only for Slave_TIMER
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the Master_TIMER and Slave_TIMER DMA request */
```

```
hrtimer_dmamode_config(HRTIMER0, HRTIMER_MASTER_TIMER,
HRTIMER_DMAMODE_NONE);
```

## hrtimer\_bunchmode\_struct\_para\_init

The description of hrtimer\_bunchmode\_struct\_para\_init is shown as below:

**Table 3-1026. Function hrtimer\_bunchmode\_struct\_para\_init**

<b>Function name</b>	hrtimer_bunchmode_struct_para_init
<b>Function prototype</b>	void hrtimer_bunchmode_struct_para_init(hrtimer_bunchmode_parameter_struct * bmcfg);
<b>Function descriptions</b>	initialize bunch mode configuration struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>bmcfg</b>	hrtimer_bunchmode_parameter_struct, the structure members can refer to <a href="#">Table 3-952. Structure hrtimer_bunchmode_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize bunch mode configuration struct with a default value */
hrtimer_bunchmode_parameter_struct bmcfg;
hrtimer_bunchmode_struct_para_init(&bmcfg);
```

## hrtimer\_bunchmode\_config

The description of hrtimer\_bunchmode\_config is shown as below:

**Table 3-1027. Function hrtimer\_bunchmode\_config**

<b>Function name</b>	hrtimer_bunchmode_config
<b>Function prototype</b>	void hrtimer_bunchmode_config(uint32_t hrtimer_periph, hrtimer_bunchmode_parameter_struct* bmcfg);
<b>Function descriptions</b>	configure bunch mode for the HRTIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>bmcfg</b>	hrtimer_bunchmode_parameter_struct, the structure members can refer to <a href="#">Table 3-952. Structure hrtimer_bunchmode_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure bunch mode for the HRTIMER */

hrtimer_bunchmode_parameter_struct bmcfg;

hrtimer_bunchmode_struct_para_init(&bmcfg);

bmcfg.clock_source = HRTIMER_BUNCHMODE_CLOCKSOURCE_ST0;

bmcfg.idle_duration = 3;

bmcfg.mode = HRTIMER_BUNCHMODE_CONTINUOUS;

bmcfg.period = 6;

bmcfg.prescaler = HRTIMER_BUNCHMODE_PRESCALER_DIV1;

bmcfg.shadow = HRTIMER_BUNCHMODEPRELOAD_DISABLED;

bmcfg.trigger[0] = HRTIMER_BUNCHMODE_TRIGGER_SOFTWARE;

bmcfg.trigger[1] = HRTIMER_BUNCHMODE_TRIGGER_NONE;

hrtimer_bunchmode_config(HRTIMER0, &bmcfg);

```

### **hrtimer\_bunchmode\_enable**

The description of hrtimer\_bunchmode\_enable is shown as below:

**Table 3-1028. Function hrtimer\_bunchmode\_enable**

Function name	hrtimer_bunchmode_enable
Function prototype	void hrtimer_bunchmode_enable(uint32_t hrtimer_periph);
Function descriptions	enable bunch mode for the HRTIMER
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable bunch mode for the HRTIMER */

hrtimer_bunchmode_enable(HRTIMER0);

```

## hrtimer\_bunchmode\_disable

The description of hrtimer\_bunchmode\_disable is shown as below:

**Table 3-1029. Function hrtimer\_bunchmode\_disable**

<b>Function name</b>	hrtimer_bunchmode_disable
<b>Function prototype</b>	void hrtimer_bunchmode_disable(uint32_t hrtimer_periph);
<b>Function descriptions</b>	disable bunch mode for the HRTIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable bunch mode for the HRTIMER */
```

```
hrtimer_bunchmode_disable(HRTIMER0);
```

## hrtimer\_bunchmode\_flag\_get

The description of hrtimer\_bunchmode\_flag\_get is shown as below:

**Table 3-1030. Function hrtimer\_bunchmode\_flag\_get**

<b>Function name</b>	hrtimer_bunchmode_flag_get
<b>Function prototype</b>	uint32_t hrtimer_bunchmode_flag_get(uint32_t hrtimer_periph);
<b>Function descriptions</b>	get bunch mode operating flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	HRTIMER_BUNCHMODE_OPERATION_OFF or HRTIMER_BUNCHMODE_OPERATION_ON

Example:

```
/* get bunch mode operating flag */
```

```
uint32_t flag;
```

```
flag = hrtimer_bunchmode_flag_get(HRTIMER0);
```

### hrtimer\_bunchmode\_software\_start

The description of hrtimer\_bunchmode\_software\_start is shown as below:

**Table 3-1031. Function hrtimer\_bunchmode\_software\_start**

<b>Function name</b>	hrtimer_bunchmode_software_start
<b>Function prototype</b>	void hrtimer_bunchmode_software_start(uint32_t hrtimer_periph);
<b>Function descriptions</b>	bunch mode started by software
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* bunch mode started by software */
```

```
hrtimer_bunchmode_software_start(HRTIMER0);
```

### hrtimer\_slavetimer\_capture\_config

The description of hrtimer\_slavetimer\_capture\_config is shown as below:

**Table 3-1032. Function hrtimer\_slavetimer\_capture\_config**

<b>Function name</b>	hrtimer_slavetimer_capture_config
<b>Function prototype</b>	void hrtimer_slavetimer_capture_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t capturex, uint64_t trgsource);
<b>Function descriptions</b>	configure the capture source in Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)



Input parameter{in}	
<b>capturex</b>	capture unit index
<i>HRTIMER_CAPTURE_ y(y=0,1)</i>	capture unit selection(y=0,1)
Input parameter{in}	
<b>trgsource</b>	capture trigger source
<i>HRTIMER_CAPTURET RIGGER_NONE</i>	Capture trigger is disabled
<i>HRTIMER_CAPTURET RIGGER_UPDATE</i>	capture triggered by update event
<i>HRTIMER_CAPTURET RIGGER_EXEV_y(y=0.. .9)</i>	capture triggered by external event 0(y=0..9)
<i>HRTIMER_CAPTURET RIGGER_STy_ACTIVE (y=0..7)</i>	capture triggered by STyCH0_O output inactive to active transition(y=0..7)
<i>HRTIMER_CAPTURET RIGGER_STy_INACTI VE(y=0..7)</i>	capture triggered by STyCH0_O output active to inactive transition(y=0..7)
<i>HRTIMER_CAPTURET RIGGER_STy_CMP0(y =0..7)</i>	capture triggered by compare 0 event of Slave_TIMERy(y=0..7)
<i>HRTIMER_CAPTURET RIGGER_STy_CMP1(y =0..7)</i>	capture triggered by compare 1 event of Slave_TIMERy(y=0..7)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the capture source in Slave_TIMER */
```

```
hrtimer_slavetimer_capture_config(HRTIMER0, HRTIMER_SLAVE_TIMER0,  
HRTIMER_CAPTURE_0, HRTIMER_CAPTURETRIGGER_NONE);
```

### **hrtimer\_slavetimer\_capture\_software**

The description of hrtimer\_slavetimer\_capture\_software is shown as below:

**Table 3-1033. Function hrtimer\_slavetimer\_capture\_software**

<b>Function name</b>	hrtimer_slavetimer_capture_software
<b>Function prototype</b>	void hrtimer_slavetimer_capture_software(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t capturex);

<b>Function descriptions</b>	configure the capture source in Slave_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)
<b>Input parameter{in}</b>	
<b>capturex</b>	capture unit index
<i>HRTIMER_CAPTURE_y(y=0,1)</i>	capture unit selection(y=0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the capture source in Slave_TIMER */
```

```
hrtimer_slavetimer_capture_software(HRTIMER0, HRTIMER_SLAVE  
HRTIMER_CAPTURE_0);
```

### **hrtimer\_slavetimer\_capture\_value\_read**

The description of hrtimer\_slavetimer\_capture\_value\_read is shown as below:

**Table 3-1034. Function hrtimer\_slavetimer\_capture\_value\_read**

<b>Function name</b>	hrtimer_slavetimer_capture_value_read
<b>Function prototype</b>	hrtimer_capture_value_struct hrtimer_slavetimer_capture_value_read(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t capturex);
<b>Function descriptions</b>	read the capture value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)

Input parameter{in}	
<b>capturex</b>	capture unit index
<i>HRTIMER_CAPTURE_</i> <i>y(y=0,1)</i>	capture unit selection(y=0,1)
Output parameter{out}	
-	-
Return value	
<b>uint32_t</b>	capture value and capture direction

Example:

```
/* read the capture value */
```

```
hrtimer_capture_value_struct captured;
```

```
captured = hrtimer_slavetimer_capture_value_read (HRTIMER0, HRTIMER_SLAVE  
HRTIMER_CAPTURE_0);
```

### hrtimer\_adctrigcfg\_struct\_para\_init

The description of hrtimer\_adctrigcfg\_struct\_para\_init is shown as below:

**Table 3-1035. Function hrtimer\_adctrigcfg\_struct\_para\_init**

<b>Function name</b>	hrtimer_adctrigcfg_struct_para_init
<b>Function prototype</b>	void hrtimer_adctrigcfg_struct_para_init(hrtimer_adctrigcfg_parameter_struct* triggercfg);
<b>Function descriptions</b>	initialize ADC trigger configuration struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>triggercfg</b>	hrtimer_adctrigcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-956. Structure hrtimer_adctrigcfg_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize ADC trigger configuration struct with a default value */
```

```
hrtimer_adctrigcfg_parameter_struct triggercfg;
```

```
hrtimer_adctrigcfg_struct_para_init(&triggercfg);
```

## hrtimer\_adc\_trigger0\_3\_config

The description of hrtimer\_adc\_trigger0\_3\_config is shown as below:

**Table 3-1036. Function hrtimer\_adc\_trigger0\_3\_config**

<b>Function name</b>	hrtimer_adc_trigger0_3_config
<b>Function prototype</b>	void hrtimer_adc_trigger0_3_config(uint32_t hrtimer_periph, uint32_t trigger_id, hrtimer_adctrigcfg_parameter_struct* triggercfg);
<b>Function descriptions</b>	configure the trigger source to ADC and the update source(0..3)
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>trigger_id</b>	ADC trigger event
<i>HRTIMER_ADCTRIG_y</i> (y=0..3)	the slave timer selection(x=0..7)
<b>Input parameter{in}</b>	
<b>triggercfg</b>	hrtimer_adctrigcfg_parameter_struct, the structure members can refer to <a href="#">Table 3-956. Structure hrtimer_adctrigcfg_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure the trigger source to ADC and the update source */
hrtimer_adctrigcfg_parameter_struct triggercfg;
hrtimer_adctrigcfg_struct_para_init(&triggercfg);
triggercfg.update_source = HRTIMER_ADCTRG1_UPDATE_MT;
triggercfg.Trigger[0] = HRTIMER_ADCTRG102_EVENT_MTCMP0;
triggercfg.Trigger[1] = HRTIMER_ADCTRG102_EVENT_NONE;

hrtimer_adc_trigger_config(HRTIMER0, HRTIMER_ADCTRIG_0, &triggercfg);

```

## hrtimer\_adc\_trigger4\_9\_config

The description of hrtimer\_adc\_trigger4\_9\_config is shown as below:

Table 3-1037. Function `hrtimer_adc_trigger_config`

Function name	<code>hrtimer_adc_trigger4_9_config</code>
Function prototype	<code>void hrtimer_adc_trigger4_9_config(uint32_t hrtimer_periph, uint32_t trigger_id, hrtimer_adctrigcfg_parameter_struct* triggercfg);</code>
Function descriptions	configure the trigger source to ADC and the update source(0..3)
Precondition	-
The called functions	-
Input parameter{in}	
<code>hrtimer_periph</code>	HRTIMER peripheral
<code>HRTIMERx(x=0)</code>	HRTIMER selection
Input parameter{in}	
<code>trigger_id</code>	ADC trigger event
<code>HRTIMER_ADCTRIG_y</code> (y=4..9)	the slave timer selection(x=0..7)
Input parameter{in}	
<code>triggercfg</code>	<code>hrtimer_adctrigcfg_parameter_struct</code> , the structure members can refer to <a href="#">Table 3-956. Structure <code>hrtimer_adctrigcfg_parameter_struct</code></a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure the trigger source to ADC and the update source */

hrtimer_adctrigcfg_parameter_struct triggercfg;

hrtimer_adctrigcfg_struct_para_init(&triggercfg);

triggercfg.update_source = HRTIMER_ADCTRIG_UPDATE_MT;

triggercfg.Trigger4_9 = HRTIMER_ADC_EXT_TRIG_579_BY_MASTER_PERIOD;

hrtimer_adc_trigger_config(HRTIMER0, HRTIMER_ADCTRIG_0, &triggercfg);

```

### `hrtimer_adc_prescaler_config`

The description of `hrtimer_adc_prescaler_config` is shown as below:

Table 3-1038. Function `hrtimer_adc_prescaler_config`

Function name	<code>hrtimer_adc_prescaler_config</code>
Function prototype	<code>void hrtimer_adc_prescaler_config(uint32_t hrtimer_periph, uint32_t trigger_id, uint32_t psc);</code>
Function descriptions	configure the trigger source prescaler of ADC (0..9)
Precondition	-
The called functions	-

Input parameter{in}	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
Input parameter{in}	
<b>timer_id</b>	HRTIMER peripheral
<i>HRTIMER_MASTER_TIMER</i>	the counter of Master_TIMER
<i>HRTIMER_SLAVE_TIMERx(x=0..7)</i>	the counter of Slave_TIMERx(x=0..7)
Input parameter{in}	
<b>psc</b>	ADC trigger prescaler value (0x00~0x1F)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the trigger source prescaler of ADC trigger 0 */
```

```
hrtimer_adc_prescaler_config (HRTIMER0, HRTIMER_ADCTRIG_0, 100);
```

### **hrtimer\_slavetimer\_counter\_direction\_get**

The description of hrtimer\_slavetimer\_counter\_direction\_get is shown as below:

**Table 3-1039. Function hrtimer\_slavetimer\_counter\_direction\_get**

<b>Function name</b>	hrtimer_slavetimer_counter_direction_get
<b>Function prototype</b>	DirectionStatus hrtimer_slavetimer_counter_direction_get(uint32_t hrtimer_periph, uint32_t timer_id);
<b>Function descriptions</b>	get the Slave_TIMER counter direction
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
Input parameter{in}	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)
Output parameter{out}	
-	-
Return value	
<b>DirectionStatus</b>	COUNTER_UP or COUNTER_DOWN

Example:

```
/* get the Slave_TIMER counter direction */
```

```
DirectionStatus dir;
```

```
dir = hrtimer_slavetimer_counter_direction_get(HRTIMER0, HRTIMER_SLAVE_TIMER0);
```

### hrtimer\_timers\_flag\_get

The description of hrtimer\_timers\_flag\_get is shown as below:

**Table 3-1040. Function hrtimer\_timers\_flag\_get**

Function name	hrtimer_timers_flag_get
Function prototype	FlagStatus hrtimer_timers_flag_get(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t flag);
Function descriptions	get the Master_TIMER and Slave_TIMER flag
Precondition	-
The called functions	-
Input parameter{in}	
hrtimer_periph	HRTIMER peripheral
HRTIMERx(x=0)	HRTIMER selection
Input parameter{in}	
timer_id	master timer and slave timer index
HRTIMER_MASTER_TIMER	the master timer
HRTIMER_SLAVE_TIMERx	the slave timer selection(x=0..7)
Input parameter{in}	
flag	the flag source
HRTIMER_MT_ST_FLAG_CMPy(y=0..3)	compare y flag, for Master_TIMER and Slave_TIMER (y=0..3)
HRTIMER_MT_ST_FLAG_AG_REP	repetition flag, for Master_TIMER and Slave_TIMER
HRTIMER_MT_INT_FLAG_AG_SYNI	synchronization input flag, for Master_TIMER
HRTIMER_MT_ST_FLAG_AG_UPD	update flag, for Master_TIMER and Slave_TIMER
HRTIMER_ST_FLAG_CAPy(y=0,1)	capture y flag, for Slave_TIMER(y=0,1)
HRTIMER_ST_FLAG_CHyOA(y=0,1)	channel y output active flag, for Slave_TIMER(y=0,1)
HRTIMER_ST_FLAG_CHyONA(y=0,1)	channel y output inactive flag, for Slave_TIMER(y=0,1)
HRTIMER_ST_FLAG_CNTRST	counter reset flag, for Slave_TIMER
HRTIMER_ST_FLAG_	delayed IDLE mode entry flag, for Slave_TIMER

<i>DLYIDLE</i>	
<i>HRTIMER_ST_FLAG_CBLN</i>	current balanced flag, for Slave_TIMER
<i>HRTIMER_ST_FLAG_BLNIDLE</i>	balanced IDLE flag, for Slave_TIMER
<i>HRTIMER_ST_FLAG_CHyOUTS(y=0,1)</i>	channel y output status flag, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_FLAG_CHyOUT(y=0,1)</i>	channel y output flag, for Slave_TIMER(y=0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the Master_TIMER and Slave_TIMER flag */
```

```
FlagStatus flag = RESET;
```

```
flag = hrtimer_timers_flag_get(HRTIMER0, HRTIMER_MASTER_TIMER,
HRTIMER_MT_ST_FLAG_CMP0);
```

### hrtimer\_timers\_flag\_clear

The description of hrtimer\_timers\_flag\_clear is shown as below:

**Table 3-1041. Function hrtimer\_timers\_flag\_clear**

<b>Function name</b>	hrtimer_timers_flag_clear
<b>Function prototype</b>	void hrtimer_timers_flag_clear(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t flag);
<b>Function descriptions</b>	clear the Master_TIMER and Slave_TIMER flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_MASTER_TIMER</i>	the master timer
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)
<b>Input parameter{in}</b>	
<b>flag</b>	the flag source
<i>HRTIMER_MT_ST_FL</i>	compare y flag, for Master_TIMER and Slave_TIMER (y=0..3)



AG_CMPy(y=0..3)	
HRTIMER_MT_ST_FL AG_REP	repetition flag, for Master_TIMER and Slave_TIMER
HRTIMER_MT_INT_FL AG_SYNI	synchronization input flag, for Master_TIMER
HRTIMER_MT_ST_FL AG_UPD	update flag, for Master_TIMER and Slave_TIMER
HRTIMER_ST_FLAG_ CAPy(y=0,1)	capture y flag, for Slave_TIMER(y=0,1)
HRTIMER_ST_FLAG_ CHyOA(y=0,1)	channel y output active flag, for Slave_TIMER(y=0,1)
HRTIMER_ST_FLAG_ CHyONA(y=0,1)	channel y output inactive flag, for Slave_TIMER(y=0,1)
HRTIMER_ST_FLAG_ CNTRST	counter reset flag, for Slave_TIMER
HRTIMER_ST_FLAG_ DLYIDLE	delayed IDLE mode entry flag, for Slave_TIMER
HRTIMER_ST_FLAG_ CBLN	current balanced flag, for Slave_TIMER
HRTIMER_ST_FLAG_ BLNIDLE	balanced IDLE flag, for Slave_TIMER
HRTIMER_ST_FLAG_ CHyOUTS(y=0,1)	channel y output state flag, for Slave_TIMER(y=0,1)
HRTIMER_ST_FLAG_ CHyOUT(y=0,1)	channel y output flag, for Slave_TIMER(y=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* clear the Master_TIMER and Slave_TIMER flag */

hrtimer_timers_flag_clear(HRTIMER0, HRTIMER_MASTER_TIMER,
HRTIMER_MT_ST_FLAG_CMP0);

```

### hrtimer\_common\_flag\_get

The description of hrtimer\_common\_flag\_get is shown as below:

**Table 3-1042. Function hrtimer\_common\_flag\_get**

Function name	hrtimer_common_flag_get
Function prototype	FlagStatus hrtimer_common_flag_get(uint32_t hrtimer_periph, uint32_t flag);

<b>Function descriptions</b>	get the common flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>flag</b>	the flag source
<i>HRTIMER_FLAG_FLTy</i> (y=0..7)	fault y interrupt flag (y=0..7)
<i>HRTIMER_FLAG_SYS</i> <i>FLT</i>	system fault interrupt flag
<i>HRTIMER_FLAG_DLL</i> <i>CAL</i>	DLL calibration completed interrupt flag
<i>HRTIMER_FLAG_BMP</i> <i>ER</i>	bunch mode period interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the common flag */
```

```
FlagStatus flag = RESET;
```

```
flag = hrtimer_common_flag_get(HRTIMER0, HRTIMER_FLAG_FLT0);
```

### hrtimer\_common\_flag\_clear

The description of hrtimer\_common\_flag\_clear is shown as below:

**Table 3-1043. Function hrtimer\_common\_flag\_clear**

<b>Function name</b>	hrtimer_common_flag_clear
<b>Function prototype</b>	void hrtimer_common_flag_clear(uint32_t hrtimer_periph, uint32_t flag);
<b>Function descriptions</b>	clear the common flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>flag</b>	the flag source
<i>HRTIMER_FLAG_FLTy</i>	fault y interrupt flag (y=0..7)

(y=0..7)	
<i>HRTIMER_FLAG_SYS FLT</i>	system fault interrupt flag
<i>HRTIMER_FLAG_DLL CAL</i>	DLL calibration completed interrupt flag
<i>HRTIMER_FLAG_BMP ER</i>	bunch mode period interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the common flag */
```

```
hrtimer_common_flag_clear(HRTIMER0, HRTIMER_FLAG_FLT0);
```

### **hrtimer\_timers\_interrupt\_enable**

The description of `hrtimer_timers_interrupt_enable` is shown as below:

**Table 3-1044. Function `hrtimer_timers_interrupt_enable`**

<b>Function name</b>	<code>hrtimer_timers_interrupt_enable</code>
<b>Function prototype</b>	<code>void hrtimer_timers_interrupt_enable(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t interrupt);</code>
<b>Function descriptions</b>	enable the Master_TIMER and Slave_TIMER interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_MASTER_TIMER</i>	the master timer
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)
<b>Input parameter{in}</b>	
<b>interrupt</b>	the interrupt source
<i>HRTIMER_MT_ST_INT _CMPy(y=0..3)</i>	compare y interrupt, for Master_TIMER and Slave_TIMER (y=0..7)
<i>HRTIMER_MT_ST_INT _REP</i>	repetition interrupt, for Master_TIMER and Slave_TIMER
<i>HRTIMER_MT_INT_SY</i>	synchronization input interrupt, for Master_TIMER

<i>NI</i>	
<i>HRTIMER_MT_ST_INT_UPD</i>	update interrupt, for Master_TIMER and Slave_TIMER
<i>HRTIMER_ST_INT_CA</i> <i>Py(y=0,1)</i>	capture y interrupt, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_CH</i> <i>yOA(y=0,1)</i>	channel y output active interrupt, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_CH</i> <i>yONA(y=0,1)</i>	channel y output inactive interrupt, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_CN</i> <i>TRST</i>	counter reset interrupt, for Slave_TIMER
<i>HRTIMER_ST_INT_DL</i> <i>YIDLE</i>	delayed IDLE mode entry interrupt, for Slave_TIMER
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the Master_TIMER and Slave_TIMER interrupt */
```

```
hrtimer_timers_interrupt_enable(HRTIMER0, HRTIMER_MASTER_TIMER,  
HRTIMER_MT_ST_INT_CMP0);
```

### **hrtimer\_timers\_interrupt\_disable**

The description of `hrtimer_timers_interrupt_disable` is shown as below:

**Table 3-1045. Function `hrtimer_timers_interrupt_disable`**

<b>Function name</b>	<code>hrtimer_timers_interrupt_disable</code>
<b>Function prototype</b>	<code>void hrtimer_timers_interrupt_disable(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t interrupt);</code>
<b>Function descriptions</b>	disable the Master_TIMER and Slave_TIMER interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_MASTER_TIMER</i>	the master timer
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)

Input parameter{in}	
<b>interrupt</b>	the interrupt source
<i>HRTIMER_MT_ST_INT_CMPy(y=0..3)</i>	compare y interrupt, for Master_TIMER and Slave_TIMER (y=0..3)
<i>HRTIMER_MT_ST_INT_REP</i>	repetition interrupt, for Master_TIMER and Slave_TIMER
<i>HRTIMER_MT_INT_SYNC</i>	synchronization input interrupt, for Master_TIMER
<i>HRTIMER_MT_ST_INT_UPD</i>	update interrupt, for Master_TIMER and Slave_TIMER
<i>HRTIMER_ST_INT_CAPy(y=0,1)</i>	capture y interrupt, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_CHyOA(y=0,1)</i>	channel y output active interrupt, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_CHyONA(y=0,1)</i>	channel y output inactive interrupt, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_CNTRST</i>	counter reset interrupt, for Slave_TIMER
<i>HRTIMER_ST_INT_DLYIDLE</i>	delayed IDLE mode entry interrupt, for Slave_TIMER
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the Master_TIMER and Slave_TIMER interrupt */
```

```
hrtimer_timers_interrupt_disable(HRTIMER0, HRTIMER_MASTER_TIMER,
HRTIMER_MT_ST_INT_CMP0);
```

### hrtimer\_timers\_interrupt\_flag\_get

The description of hrtimer\_timers\_interrupt\_flag\_get is shown as below:

**Table 3-1046. Function hrtimer\_timers\_interrupt\_flag\_get**

<b>Function name</b>	hrtimer_timers_interrupt_flag_get
<b>Function prototype</b>	FlagStatus hrtimer_timers_interrupt_flag_get(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t interrupt);
<b>Function descriptions</b>	get the Master_TIMER and Slave_TIMER interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>hrtimer_periph</b>	HRTIMER peripheral

<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_MASTER_TIMER</i>	the master timer
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)
<b>Input parameter{in}</b>	
<b>interrupt</b>	the interrupt source
<i>HRTIMER_MT_ST_INT_FLAG_CMPy(y=0..3)</i>	compare y interrupt flag, for Master_TIMER and Slave_TIMER (y=0..3)
<i>HRTIMER_MT_ST_INT_FLAG_REP</i>	repetition interrupt flag, for Master_TIMER and Slave_TIMER
<i>HRTIMER_MT_INT_FLAG_SYNC</i>	synchronization input interrupt flag, for Master_TIMER
<i>HRTIMER_MT_ST_INT_FLAG_UPD</i>	update interrupt flag, for Master_TIMER and Slave_TIMER
<i>HRTIMER_ST_INT_FLAG_CAPy(y=0,1)</i>	capture y interrupt flag, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_FLAG_CHyOA(y=0,1)</i>	channel y output active interrupt flag, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_FLAG_CHyONA(y=0,1)</i>	channel y output inactive interrupt flag, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_FLAG_CNTRST</i>	counter reset interrupt flag, for Slave_TIMER
<i>HRTIMER_ST_INT_FLAG_DLYIDLE</i>	delayed IDLE mode entry interrupt flag, for Slave_TIMER
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the Master_TIMER and Slave_TIMER interrupt flag */
```

```
FlagStatus flag = RESET;
```

```
flag = hrtimer_timers_interrupt_flag_get(HRTIMER0, HRTIMER_MASTER_TIMER,
HRTIMER_MT_ST_INT_FLAG_CMP0);
```

### **hrtimer\_timers\_interrupt\_flag\_clear**

The description of `hrtimer_timers_interrupt_flag_clear` is shown as below:

Table 3-1047. Function `hrtimer_timers_interrupt_flag_clear`

<b>Function name</b>	<code>hrtimer_timers_interrupt_flag_clear</code>
<b>Function prototype</b>	<code>void hrtimer_timers_interrupt_flag_clear(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t interrupt);</code>
<b>Function descriptions</b>	clear the Master_TIMER and Slave_TIMER interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>timer_id</b>	master timer and slave timer index
<i>HRTIMER_MASTER_TIMER</i>	the master timer
<i>HRTIMER_SLAVE_TIMERx</i>	the slave timer selection(x=0..7)
<b>Input parameter{in}</b>	
<b>interrupt</b>	the interrupt source
<i>HRTIMER_MT_ST_INT_FLAG_CMPy(y=0..3)</i>	compare y interrupt flag, for Master_TIMER and Slave_TIMER (y=0..3)
<i>HRTIMER_MT_ST_INT_FLAG_REP</i>	repetition interrupt flag, for Master_TIMER and Slave_TIMER
<i>HRTIMER_MT_INT_FLAG_SYNC</i>	synchronization input interrupt flag, for Master_TIMER
<i>HRTIMER_MT_ST_INT_FLAG_UPD</i>	update interrupt flag, for Master_TIMER and Slave_TIMER
<i>HRTIMER_ST_INT_FLAG_CAPy(y=0,1)</i>	capture y interrupt flag, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_FLAG_CHyOA(y=0,1)</i>	channel y output active interrupt flag, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_FLAG_CHyONA(y=0,1)</i>	channel y output inactive interrupt flag, for Slave_TIMER(y=0,1)
<i>HRTIMER_ST_INT_FLAG_CNTRST</i>	counter reset interrupt flag, for Slave_TIMER
<i>HRTIMER_ST_INT_FLAG_DLYIDLE</i>	delayed IDLE mode entry interrupt flag, for Slave_TIMER
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the Master_TIMER and Slave_TIMER interrupt flag */
```

```
hrtimer_timers_interrupt_flag_clear(HRTIMER0, HRTIMER_MASTER_TIMER,
HRTIMER_MT_ST_INT_FLAG_CMP0);
```

### hrtimer\_common\_interrupt\_enable

The description of hrtimer\_common\_interrupt\_enable is shown as below:

**Table 3-1048. Function hrtimer\_common\_interrupt\_enable**

<b>Function name</b>	hrtimer_common_interrupt_enable
<b>Function prototype</b>	void hrtimer_common_interrupt_enable(uint32_t hrtimer_periph, uint32_t interrupt);
<b>Function descriptions</b>	enable HRTIMER common interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx(x=0)</i>	HRTIMER selection
<b>Input parameter{in}</b>	
<b>interrupt</b>	the interrupt source
<i>HRTIMER_INT_FLTy(y=0..7)</i>	fault y interrupt (y=0..7)
<i>HRTIMER_INT_SYSFLT</i>	system fault interrupt
<i>HRTIMER_INT_DLLCAL</i>	DLL calibration completed interrupt
<i>HRTIMER_INT_BMPE</i>	bunch mode period interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable HRTIMER common interrupt */
hrtimer_common_interrupt_enable(HRTIMER0, HRTIMER_INT_FLT0);
```

### hrtimer\_common\_interrupt\_disable

The description of hrtimer\_common\_interrupt\_disable is shown as below:

**Table 3-1049. Function hrtimer\_common\_interrupt\_disable**

<b>Function name</b>	hrtimer_common_interrupt_disable
<b>Function prototype</b>	void hrtimer_common_interrupt_disable(uint32_t hrtimer_periph, uint32_t interrupt);



<b>Function descriptions</b>	disable HRTIMER common interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx</i> (x=0)	HRTIMER selection
<b>Input parameter{in}</b>	
<b>interrupt</b>	the interrupt source
<i>HRTIMER_INT_FLTy</i> (y=0..7)	fault y interrupt (y=0..7)
<i>HRTIMER_INT_SYSFLT</i>	system fault interrupt
<i>HRTIMER_INT_DLLCAL</i>	DLL calibration completed interrupt
<i>HRTIMER_INT_BMPER</i>	bunch mode period interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable HRTIMER common interrupt */
```

```
hrtimer_common_interrupt_disable(HRTIMER0, HRTIMER_INT_FLT0);
```

### hrtimer\_common\_interrupt\_flag\_get

The description of hrtimer\_common\_interrupt\_flag\_get is shown as below:

**Table 3-1050. Function hrtimer\_common\_interrupt\_flag\_get**

<b>Function name</b>	hrtimer_common_interrupt_flag_get
<b>Function prototype</b>	FlagStatus hrtimer_common_interrupt_flag_get(uint32_t hrtimer_periph, uint32_t interrupt);
<b>Function descriptions</b>	get the common interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx</i> (x=0)	HRTIMER selection
<b>Input parameter{in}</b>	
<b>interrupt</b>	the interrupt source
<i>HRTIMER_INT_FLAG_FLTy</i> (y=0..7)	fault y interrupt flag (y=0..7)

<i>HRTIMER_INT_FLAG_SYSFLT</i>	system fault interrupt flag
<i>HRTIMER_INT_FLAG_DLLCAL</i>	DLL calibration completed interrupt flag
<i>HRTIMER_INT_FLAG_BMPER</i>	bunch mode period interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the common interrupt flag */
```

```
FlagStatus flag = RESET;
```

```
flag = hrtimer_common_interrupt_flag_get(HRTIMER0, HRTIMER_INT_FLAG_FLT0);
```

### **hrtimer\_common\_interrupt\_flag\_clear**

The description of `hrtimer_common_interrupt_flag_clear` is shown as below:

**Table 3-1051. Function `hrtimer_common_interrupt_flag_clear`**

<b>Function name</b>	<code>hrtimer_common_interrupt_flag_clear</code>
<b>Function prototype</b>	<code>void hrtimer_common_interrupt_flag_clear(uint32_t hrtimer_periph, uint32_t interrupt);</code>
<b>Function descriptions</b>	clear the common interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hrtimer_periph</b>	HRTIMER peripheral
<i>HRTIMERx</i> ( <i>x</i> =0)	HRTIMER selection
<b>Input parameter{in}</b>	
<b>interrupt</b>	the interrupt source
<i>HRTIMER_INT_FLAG_FLTy</i> ( <i>y</i> =0..7)	fault y interrupt flag ( <i>y</i> =0..7)
<i>HRTIMER_INT_FLAG_SYSFLT</i>	system fault interrupt flag
<i>HRTIMER_INT_FLAG_DLLCAL</i>	DLL calibration completed interrupt flag
<i>HRTIMER_INT_FLAG_BMPER</i>	bunch mode period interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* clear the common interrupt flag */
```

```
hrtimer_common_interrupt_flag_clear(HRTIMER0, HRTIMER_INT_FLAG_FLT0);
```

## 3.28. SPI

The SPI module can communicate with external devices using the SPI protocol. The SPI registers are listed in chapter [3.28.1](#), the SPI firmware functions are introduced in chapter [3.28.2](#).

### 3.28.1. Descriptions of Peripheral registers

SPI registers are listed in the table shown as below:

**Table 3-1052. SPI 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_QCTL	Quad-SPI mode control register

### 3.28.2. Descriptions of Peripheral functions

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

**Table 3-1053. SPI firmware function**

Function name	Function description
spi_deinit	reset SPI peripheral
spi_struct_para_init	initialize the parameters of SPI struct
spi_init	initialize SPI parameter
spi_enable	enable SPI
spi_disable	disable SPI
spi_nss_output_enable	enable SPI NSS output function
spi_nss_output_disable	disable SPI NSS output function
spi_nss_internal_high	SPI NSS pin high level in software mode
spi_nss_internal_low	SPI NSS pin low level in software mode
spi_dma_enable	enable SPI DMA function

Function name	Function description
spi_dma_disable	disable SPI DMA function
spi_transmit_odd_config	configure SPI total number of data transmitting by DMA is odd or not
spi_receive_odd_config	configure SPI total number of data receiving by DMA is odd or not
spi_data_frame_format_config	configure SPI data frame format
spi_fifo_access_size_config	configure SPI access size to FIFO(8-bit or 16-bit)
spi_bidirectional_transfer_config	configure SPI bidirectional transfer direction
spi_data_transmit	SPI transmit data
spi_data_receive	SPI receive data
spi_crc_polynomial_set	set SPI CRC polynomial
spi_crc_polynomial_get	get SPI CRC polynomial
spi_crc_length_set	set CRC length
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_format_error_clear	clear SPI format error flag status
spi_flag_get	get SPI flag status
spi_interrupt_enable	enable SPI interrupt
spi_interrupt_disable	disable SPI interrupt
spi_interrupt_flag_get	get SPI interrupt status

## Structure spi\_parameter\_struct

**Table 3-1054. 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_CLOCK_PL_LOW_PH_1EDGE, SPI_CLOCK_PL_HIGH_PH_1EDGE, SPI_CLOCK_PL_LOW_PH_2EDGE, SPI_CLOCK_PL_HIGH_PH_2EDGE)
prescale	SPI prescale factor (SPI_PSC_n (n=2,4,8,16,32,64,128,256))

## spi\_deinit

The description of spi\_deinit is shown as below:

**Table 3-1055. Function spi\_deinit**

<b>Function name</b>	spi_deinit
<b>Function prototype</b>	void spi_deinit(uint32_t spi_periph);
<b>Function descriptions</b>	reset SPI peripheral
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
spi_periph	SPI peripheral
SPIx	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset SPI0 */
spi_deinit(SPI0);
```

## spi\_struct\_para\_init

The description of spi\_struct\_para\_init is shown as below:

**Table 3-1056. Function spi\_struct\_para\_init**

<b>Function name</b>	spi_struct_para_init
<b>Function prototype</b>	void spi_struct_para_init(spi_parameter_struct* spi_struct);
<b>Function descriptions</b>	initialize the parameters of SPI struct with the default values
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
spi_struct	SPI parameter struct, the structure members can refer to members of the structure <a href="#">Table 3-1054. spi_parameter_struct</a>
Return value	
-	-

Example:

```

/* initialize the parameters of SPI */

spi_parameter_struct spi_struct;

spi_struct->device_mode = SPI_SLAVE;

spi_struct->trans_mode = SPI_TRANSMODE_FULLDUPLEX;

spi_struct->frame_size = SPI_FRAME_SIZE_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_struct_para_init(&spi_init_struct);

```

## spi\_init

The description of spi\_init is shown as below:

**Table 3-1057. Function spi\_init**

Function name	spi_init
Function prototype	void spi_init(uint32_t spi_periph, spi_parameter_struct* spi_struct);
Function descriptions	initialize SPI peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SP/x	x=0,1,2
Input parameter{in}	
spi_struct	SPI parameter initialization struct, the structure members can refer to members of the structure <a href="#">Table 3-1054. spi_parameter_struct</a>
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 3-1058. Function spi\_enable**

<b>Function name</b>	spi_enable
<b>Function prototype</b>	void spi_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* enable SPI0 */

spi_enable(SPI0);

```

## spi\_disable

The description of spi\_disable is shown as below:

**Table 3-1059. Function spi\_disable**

<b>Function name</b>	spi_disable
<b>Function prototype</b>	void spi_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPIx
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 */
spi_disable(SPI0);
```

## spi\_nss\_output\_enable

The description of spi\_nss\_output\_enable is shown as below:

**Table 3-1060. Function spi\_nss\_output\_enable**

<b>Function name</b>	spi_nss_output_enable
<b>Function prototype</b>	void spi_nss_output_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI NSS output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1061. Function spi\_nss\_output\_disable**

<b>Function name</b>	spi_nss_output_disable
<b>Function prototype</b>	void spi_nss_output_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI NSS output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1062. Function spi\_nss\_internal\_high**

<b>Function name</b>	spi_nss_internal_high
<b>Function prototype</b>	void spi_nss_internal_high(uint32_t spi_periph);
<b>Function descriptions</b>	SPI NSS pin high level in software mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1063. Function spi\_nss\_internal\_low**

<b>Function name</b>	spi_nss_internal_low
<b>Function prototype</b>	void spi_nss_internal_low(uint32_t spi_periph);
<b>Function descriptions</b>	SPI NSS pin low level in software mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1064. Function spi\_dma\_enable**

<b>Function name</b>	spi_dma_enable
<b>Function prototype</b>	void spi_dma_enable(uint32_t spi_periph, uint8_t spi_dma);
<b>Function descriptions</b>	enable SPI DMA send or receive
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>spi_dma</b>	SPI DMA mode
<i>SPI_DMA_TRANSMIT</i>	SPI transmit data use DMA
<i>SPI_DMA_RECEIVE</i>	SPI receive data use DMA
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1065. Function spi\_dma\_disable**

<b>Function name</b>	spi_dma_disable
<b>Function prototype</b>	void spi_dma_disable(uint32_t spi_periph, uint8_t spi_dma);
<b>Function descriptions</b>	disable SPI DMA function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>spi_dma</b>	SPI DMA mode
<i>SPI_DMA_TRANSMIT</i>	SPI transmit data use DMA
<i>SPI_DMA_RECEIVE</i>	SPI receive data use DMA
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 transmit data DMA function */

spi_dma_disable(SPI0, SPI_DMA_TRANSMIT);
```

### spi\_transmit\_odd\_config

The description of spi\_transmit\_odd\_config is shown as below:

**Table 3-1066. Function spi\_transmit\_odd\_config**

<b>Function name</b>	spi_transmit_odd_config
<b>Function prototype</b>	void spi_transmit_odd_config(uint32_t spi_periph, uint16_t odd);
<b>Function descriptions</b>	configure SPI0 total number of data to transmit by DMA is odd or not
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2

Input parameter{in}	
<b>dma</b>	SPI DMA mode
<i>SPI_TXDMA_EVEN</i>	number of byte in TX DMA channel is even
<i>SPI_TXDMA_ODD</i>	number of byte in TX DMA channel is odd
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SPI0 total number of data to transmit by DMA is odd */
```

```
spi_transmit_odd_config (SPI0, SPI_TXDMA_ODD);
```

### spi\_receive\_odd\_config

The description of spi\_receive\_odd\_config is shown as below:

**Table 3-1067. Function spi\_receive\_odd\_config**

<b>Function name</b>	spi_receive_odd_config
<b>Function prototype</b>	void spi_receive_odd_config(uint32_t spi_periph, uint16_t odd);
<b>Function descriptions</b>	configure SPI0 total number of data to receive by DMA is odd or not
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
Input parameter{in}	
<b>dma</b>	SPI DMA mode
<i>SPI_RXDMA_EVEN</i>	number of byte in RX DMA channel is even
<i>SPI_RXDMA_ODD</i>	number of byte in RX DMA channel is odd
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SPI0 total number of data to receive by DMA is odd */
```

```
spi_receive_odd_config (SPI0, SPI_RXDMA_ODD);
```

### spi\_data\_frame\_size\_config

The description of spi\_data\_frame\_size\_config is shown as below:

Table 3-1068. Function spi\_data\_frame\_size\_config

Function name	spi_data_frame_size_config
Function prototype	void spi_data_frame_size_config(uint32_t spi_periph, uint32_t frame_size);
Function descriptions	configure SPI data frame size
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
frame_size	SPI frame size
SPI_DATASIZE_xBIT	SPI x-bit data frame size (x=4,5,..32)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure SPI0 data frame size is 16 bits */
```

```
spi_data_frame_size_config(SPI0, SPI_DATASIZE_16BIT);
```

### spi\_fifo\_access\_size\_config

The description of spi\_fifo\_access\_size\_config is shown as below:

Table 3-1069. Function spi\_fifo\_access\_size\_config

Function name	spi_fifo_access_size_config
Function prototype	void spi_fifo_access_size_config(uint32_t spi_periph, uint16_t fifo_access_size);
Function descriptions	configure SPI0 access size to FIFO(8bit or 16bit)
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
frame_format	SPI frame size
SPI_HALFWORD_ACCESS	half-word access to FIFO
SPI_BYTE_ACCESS	byte access to FIFO
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* configure SPI0 access size half word */
```

```
spi_fifo_access_size_config (SPI0, SPI_HALFWORD_ACCESS);
```

### **spi\_bidirectional\_transfer\_config**

The description of spi\_bidirectional\_transfer\_config is shown as below:

**Table 3-1070. Function spi\_bidirectional\_transfer\_config**

<b>Function name</b>	spi_bidirectional_transfer_config
<b>Function prototype</b>	void spi_bidirectional_transfer_config(uint32_t spi_periph, uint32_t transfer_direction);
<b>Function descriptions</b>	configure SPI bidirectional transfer direction
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>transfer_direction</b>	SPI transfer direction
<i>SPI_BIDIRECTIONAL_TRANSMIT</i>	SPI work in transmit-only mode
<i>SPI_BIDIRECTIONAL_RECEIVE</i>	SPI work in receive-only mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI0 works in transmit-only mode */
```

```
spi_bidirectional_transfer_config(SPI0, SPI_BIDIRECTIONAL_TRANSMIT);
```

### **spi\_data\_transmit**

The description of spi\_data\_transmit is shown as below:

**Table 3-1071. Function spi\_data\_transmit**

<b>Function name</b>	spi_data_transmit
<b>Function prototype</b>	void spi_data_transmit(uint32_t spi_periph, uint32_t data);
<b>Function descriptions</b>	SPI transmit data
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Input parameter{in}	
data	32-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 transmit data */
```

```
spi_data_transmit(SPI0, spi0_send_array[send_n]);
```

### spi\_data\_receive

The description of spi\_data\_receive is shown as below:

**Table 3-1072. Function spi\_data\_receive**

Function name	spi_data_receive
Function prototype	uint32_t spi_data_receive(uint32_t spi_periph);
Function descriptions	SPI receive data
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
uint32_t	32-bit data

Example:

```
/* SPI0 receive data */
```

```
spi0_receive_array[receive_n] = spi_data_receive(SPI0);
```

### spi\_crc\_polynomial\_set

The description of spi\_crc\_polynomial\_set is shown as below:

**Table 3-1073. Function spi\_crc\_polynomial\_set**

Function name	spi_crc_polynomial_set
---------------	------------------------

<b>Function prototype</b>	void spi_crc_polynomial_set(uint32_t spi_periph, uint32_t crc_poly);
<b>Function descriptions</b>	set SPI CRC polynomial
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>crc_poly</b>	CRC polynomial value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1074. Function spi\_crc\_polynomial\_get**

<b>Function name</b>	spi_crc_polynomial_get
<b>Function prototype</b>	uint32_t spi_crc_polynomial_get(uint32_t spi_periph);
<b>Function descriptions</b>	get SPI CRC polynomial
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	32-bit CRC polynomial (0-0xFFFFFFFF)

Example:

```

/* get SPI0 CRC polynomial */
uint32_t crc_val;
crc_val = spi_crc_polynomial_get(SPI0);

```



## spi\_crc\_length\_set

The description of spi\_crc\_length\_set is shown as below:

**Table 3-1075. Function spi\_crc\_length\_set**

<b>Function name</b>	spi_crc_length_set
<b>Function prototype</b>	void spi_crc_length_set(uint32_t spi_periph, uint16_t crc_length);
<b>Function descriptions</b>	set CRC length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>crc_length</b>	CRC length
<i>SPI_CRC_8BIT</i>	CRC length is 8 bits
<i>SPI_CRC_16BIT</i>	CRC length is 16 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set CRC length is 8 bits*/
spi_crc_length_set(SPI0, SPI_CRC_8BIT);
```

## spi\_crc\_on

The description of spi\_crc\_on is shown as below:

**Table 3-1076. Function spi\_crc\_on**

<b>Function name</b>	spi_crc_on
<b>Function prototype</b>	void spi_crc_on(uint32_t spi_periph);
<b>Function descriptions</b>	turn on CRC function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* turn on SPI0 CRC function */
```

```
spi_crc_on(SPI0);
```

### spi\_crc\_off

The description of spi\_crc\_off is shown as below:

**Table 3-1077. Function spi\_crc\_off**

<b>Function name</b>	spi_crc_off
<b>Function prototype</b>	void spi_crc_off(uint32_t spi_periph);
<b>Function descriptions</b>	turn off CRC function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* turn off SPI0 CRC function */
```

```
spi_crc_off(SPI0);
```

### spi\_crc\_next

The description of spi\_crc\_next is shown as below:

**Table 3-1078. Function spi\_crc\_next**

<b>Function name</b>	spi_crc_next
<b>Function prototype</b>	void spi_crc_next(uint32_t spi_periph);
<b>Function descriptions</b>	SPI next data is CRC value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1079. Function spi\_crc\_get**

<b>Function name</b>	spi_crc_get
<b>Function prototype</b>	uint32_t spi_crc_get(uint32_t spi_periph, uint8_t crc);
<b>Function descriptions</b>	get SPI CRC send value or receive value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>crc</b>	SPI crc value
<i>SPI_CRC_TX</i>	get transmit CRC value
<i>SPI_CRC_RX</i>	get receive CRC value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	32-bit CRC value (0-0xFFFFFFFF)

Example:

```
/* get SPI0 CRC send value */
```

```
uint32_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 3-1080. Function spi\_crc\_error\_clear**

<b>Function name</b>	spi_crc_error_clear
<b>Function prototype</b>	void spi_crc_error_clear(uint32_t spi_periph);
<b>Function descriptions</b>	clear SPI CRC error flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear SPI 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 3-1081. Function spi\_ti\_mode\_enable**

<b>Function name</b>	spi_ti_mode_enable
<b>Function prototype</b>	void spi_ti_mode_enable (uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI TI mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI TI mode */
```

```
spi_ti_mode_enable(SPI0);
```

### spi\_ti\_mode\_disable

The description of spi\_ti\_mode\_disable is shown as below:

**Table 3-1082. Function spi\_ti\_mode\_disable**

<b>Function name</b>	spi_ti_mode_disable
<b>Function prototype</b>	void spi_ti_mode_disable (uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI TI mode
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI TI mode */
spi_ti_mode_disable(SPI0);
```

### spi\_nssp\_mode\_enable

The description of spi\_nssp\_mode\_enable is shown as below:

**Table 3-1083. Function spi\_nssp\_mode\_enable**

<b>Function name</b>	spi_nssp_mode_enable
<b>Function prototype</b>	void spi_nssp_mode_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI NSS pulse mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI NSS pulse mode */
spi_nssp_mode_enable(SPI0);
```

### spi\_nssp\_mode\_disable

The description of spi\_nssp\_mode\_disable is shown as below:

**Table 3-1084. Function spi\_nssp\_mode\_disable**

<b>Function name</b>	spi_nssp_mode_disable
<b>Function prototype</b>	void spi_nssp_mode_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI NSS pulse mode
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1,2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI NSS pulse mode */
spi_nssp_mode_disable(SPI0);
```

### spi\_quad\_enable

The description of spi\_quad\_enable is shown as below:

**Table 3-1085. Function spi\_quad\_enable**

Function name	spi_quad_enable
Function prototype	void spi_quad_enable (uint32_t spi_periph);
Function descriptions	enable quad wire SPI
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI0 quad wire mode */
spi_quad_enable(SPI0);
```

### spi\_quad\_disable

The description of spi\_quad\_disable is shown as below:

**Table 3-1086. 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

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 quad wire mode */
```

```
spi_quad_disable(SPI0);
```

### spi\_quad\_write\_enable

The description of spi\_quad\_write\_enable is shown as below:

**Table 3-1087. Function spi\_quad\_write\_enable**

<b>Function name</b>	spi_quad_write_enable
<b>Function prototype</b>	void spi_quad_write_enable (uint32_t spi_periph);
<b>Function descriptions</b>	enable quad wire SPI write
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI0 quad wire write */
```

```
spi_quad_write_enable(SPI0);
```

### spi\_quad\_read\_enable

The description of spi\_quad\_read\_enable is shown as below:

**Table 3-1088. Function spi\_quad\_read\_enable**

<b>Function name</b>	spi_quad_read_enable
<b>Function prototype</b>	void spi_quad_read_enable (uint32_t spi_periph);

<b>Function descriptions</b>	enable quad wire SPI read
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI0 quad wire read */
spi_quad_read_enable(SPI0);
```

### spi\_quad\_io23\_output\_enable

The description of spi\_quad\_io23\_output\_enable is shown as below:

**Table 3-1089. Function spi\_quad\_io23\_output\_enable**

<b>Function name</b>	spi_quad_io23_output_enable
<b>Function prototype</b>	void spi_quad_io23_output_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI_IO2 and SPI_IO3 pin output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI3 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 3-1090. Function spi\_quad\_io23\_output\_disable**

<b>Function name</b>	spi_quad_io23_output_disable
----------------------	------------------------------



<b>Function prototype</b>	void spi_quad_io23_output_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI_IO2 and SPI_IO3 pin output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI3 SPI_IO2 and SPI_IO3 pin output */
```

```
spi_quad_io23_output_disable(SPI0);
```

### spi\_format\_error\_clear

The description of spi\_format\_error\_clear is shown as below:

**Table 3-1091. Function spi\_format\_error\_clear**

<b>Function name</b>	spi_format_error_clear
<b>Function prototype</b>	void spi_format_error_clear(uint32_t spi_periph, uint32_t flag);
<b>Function descriptions</b>	clear SPI format error flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>flag</b>	SPI frame format error flag
<i>SPI_FLAG_FERR</i>	SPI format error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear SPI format error flag status */
```

```
spi_format_error_clear(SPI0, SPI_FLAG_FERR);
```

## spi\_flag\_get

The description of spi\_flag\_get is shown as below:

**Table 3-1092. Function spi\_i2s\_flag\_get**

<b>Function name</b>	spi_flag_get
<b>Function prototype</b>	FlagStatus spi_flag_get(uint32_t spi_periph, uint32_t flag);
<b>Function descriptions</b>	get SPI flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>flag</b>	SPI 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_FLAG_RXORERR</i>	receive overrun error flag
<i>SPI_FLAG_CONFERR</i>	mode config error flag
<i>SPI_FLAG_CRCERR</i>	CRC error flag
<i>SPI_FLAG_FERR</i>	SPI format error interrupt flag
only for SPI0	
<i>SPI_FLAG_TXLVL_EMPTY</i>	SPI TXFIFO is empty
<i>SPI_FLAG_TXLVL_QUARTER_FULL</i>	SPI TXFIFO is a quarter of full
<i>SPI_FLAG_TXLVL_HALF_FULL</i>	SPI TXFIFO is a half of full
<i>SPI_FLAG_TXLVL_FULL</i>	SPI TXFIFO is full
<i>SPI_FLAG_RXLVL_EMPTY</i>	SPI RXFIFO is empty
<i>SPI_FLAG_RXLVL_QUARTER_FULL</i>	SPI RXFIFO is a quarter of full
<i>SPI_FLAG_RXLVL_HALF_FULL</i>	SPI RXFIFO is a half of full
<i>SPI_FLAG_RXLVL_FULL</i>	SPI RXFIFO is full
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get SPI0 transmit buffer empty flag status */
```

```
FlagStatus Flag = RESET;
```

```
Flag = spi_flag_get(SPI0, SPI_FLAG_TBE);
```

### spi\_interrupt\_enable

The description of spi\_interrupt\_enable is shown as below:

**Table 3-1093. Function spi\_interrupt\_enable**

<b>Function name</b>	spi_interrupt_enable
<b>Function prototype</b>	void spi_interrupt_enable(uint32_t spi_periph, uint8_t interrupt);
<b>Function descriptions</b>	enable SPI interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>interrupt</b>	SPI interrupt
<i>SPI_INT_TBE</i>	transmit buffer empty interrupt
<i>SPI_INT_RBNE</i>	receive buffer not empty interrupt
<i>SPI_INT_ERR</i>	CRC error,configuration error,reception overrun error, transmission underrun error and format error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI0 transmit buffer empty interrupt */
```

```
spi_interrupt_enable(SPI0, SPI_INT_TBE);
```

### spi\_i2s\_interrupt\_disable

The description of spi\_interrupt\_disable is shown as below:

**Table 3-1094. Function spi\_interrupt\_disable**

<b>Function name</b>	spi_interrupt_disable
<b>Function prototype</b>	void spi_interrupt_disable(uint32_t spi_periph, uint8_t interrupt);
<b>Function descriptions</b>	disable SPI interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
Input parameter{in}	
<b>interrupt</b>	SPI interrupt
<i>SPI_INT_TBE</i>	transmit buffer empty interrupt
<i>SPI_INT_RBNE</i>	receive buffer not empty interrupt
<i>SPI_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_interrupt_disable(SPI0, SPI_INT_TBE);
```

### spi\_interrupt\_flag\_get

The description of spi\_interrupt\_flag\_get is shown as below:

**Table 3-1095. Function spi\_interrupt\_flag\_get**

<b>Function name</b>	spi_interrupt_flag_get
<b>Function prototype</b>	FlagStatus spi_interrupt_flag_get(uint32_t spi_periph, uint8_t interrupt);
<b>Function descriptions</b>	get SPI interrupt status
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1,2
Input parameter{in}	
<b>interrupt</b>	SPI interrupt flag status
<i>SPI_INT_FLAG_TBE</i>	transmit buffer empty interrupt
<i>SPI_INT_FLAG_RBNE</i>	receive buffer not empty interrupt
<i>SPI_INT_FLAG_RXORERR</i>	overrun interrupt
<i>SPI_INT_FLAG_CONFERR</i>	config error interrupt
<i>SPI_INT_FLAG_CRCERR</i>	CRC error interrupt
Output parameter{out}	
-	-

Return value	
FlagStatus	SET or RESET

Example:

```
/* get SPI0 transmit buffer empty interrupt status */
```

```
FlagStatus Flag_interrupt = RESET;
```

```
Flag_interrupt = spi_interrupt_flag_get(SPI0, SPI_INT_FLAG_TBE);
```

## 3.29. SYSCFG

The SYSCFG registers are listed in chapter [3.29.1](#), and the SYSCFG firmware functions are introduced in chapter [3.29.2](#).

### 3.29.1. Descriptions of Peripheral registers

SYSCFG registers are listed in the table shown as below:

**Table 3-1096. SYSCFG Registers**

Registers	Descriptions
SYSCFG_CFG0	SYSCFG configuration register 0
SYSCFG_CFG1	SYSCFG configuration register 1
SYSCFG_EXTISS0	EXTI sources selection register 0
SYSCFG_EXTISS1	EXTI sources selection register 1
SYSCFG_EXTISS2	EXTI sources selection register 2
SYSCFG_EXTISS3	EXTI sources selection register 3
SYSCFG_CFG2	SYSCFG configuration register 2
SYSCFG_STAT	SYSCFG status register
SYSCFG_CFG3	SYSCFG configuration register 3
SYSCFG_CFG4	SYSCFG configuration register 4
SYSCFG_CFG5	SYSCFG configuration register 5
SYSCFG_TCMSRAMCS	SYSCFG TCMSRAM control and status register
SYSCFG_TCMSRAMKEY	SYSCFG TCMSRAM key register
SYSCFG_TCMSRAMWP	SYSCFG TCMSRAM write protection register
SYSCFG_CPSCTL	SYSCFG I/O Compensation cell control/status register
SYSCFG_TIMERCISEL0	SYSCFG TIMER input selection register 0
SYSCFG_TIMERCISEL1	SYSCFG TIMER input selection register 1
SYSCFG_TIMERCISEL2	SYSCFG TIMER input selection register 2
SYSCFG_TIMERCISEL3	SYSCFG TIMER input selection register 3
SYSCFG_TIMERCISEL4	SYSCFG TIMER input selection register 4
SYSCFG_TIMERxCFG0, x=0, 1, 2, 3, 4, 7, 19	SYSCFG TIMER configuration register0, x=0, 1, 2, 3, 4, 7, 19

Registers	Descriptions
SYSCFG_TIMERxCFG1, x=0, 1, 2, 3, 4, 7, 19	SYSCFG TIMER configuration register1, x=0, 1, 2, 3, 4, 7, 19
SYSCFG_TIMERxCFG2, x=0, 1, 2, 3, 4, 7, 19	SYSCFG TIMER configuration register2, x=0, 1, 2, 3, 4, 7, 19
SYSCFG_TIMERxCFG0, x=14	SYSCFG TIMER configuration register0, x=14
SYSCFG_TIMERxCFG1, x=14	SYSCFG TIMER configuration register1, x=14
SYSCFG_TIMERxCFG2, x=14	SYSCFG TIMER configuration register2, x=14

### 3.29.2. Descriptions of Peripheral functions

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

**Table 3-1097. SYSCFG firmware function**

Function name	Function description
syscfg_deinit	reset the SYSCFG registers
syscfg_flash_bank_remap_set	switch flash Bank0 and Bank1 addrs
syscfg_fpu_interrupt_enable	enable FPU interrupt
syscfg_fpu_interrupt_disable	disable FPU interrupt
syscfg_i2c_fast_mode_plus_enable	enable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
syscfg_i2c_fast_mode_plus_disable	disable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
syscfg_exti_line_config	configure the GPIO pin as EXTI Line
syscfg_pin_reset_mode_config	configure GPIO reset mode
syscfg_trigsel_cla_reset_mode_config	configure Trigsel CLA reset mode
syscfg_lockup_enable	enable module lockup
syscfg_timer_input_source_select	select timer channel input source
syscfg_bootmode_memmap_select	select memory bootmode mapping
syscfg_sram_ecc_single_correctable_bit_get	SRAM ECC single correctable bit get
syscfg_sram_ecc_error_address_get	SRAM ECC error address get
syscfg_tcmsram_erase	erase tcmsram
syscfg_tcmsram_erase_lock	lock the TCMSRAM erase
syscfg_tcmsram_erase_unlock	unlock the TCMSRAM erase
syscfg_tcmsram_page_wp_enable	TCMSRAM page x write protection enable
syscfg_io_compensation_config	I/O compensation cell enable
syscfg_io_compensation_disable	I/O compensation cell disable
syscfg_interrupt_flag_get	get the interrupt flags
syscfg_interrupt_flag_clear	clear the interrupt flags

Function name	Function description
syscfg_interrupt_enable	enable the syscfg peripherals interrupt
syscfg_interrupt_disable	disable the syscfg peripherals interrupt
syscfg_tcmsram_busy_flag_get	get tcmsram erase busy flag
syscfg_compensation_cell_ready_flag_get	get compensation cell ready flag

### Enum syscfg\_interrupt\_enum

**Table 3-1098. Enum syscfg\_interrupt\_enum**

enum name	Function description
SYSCFG_INT_SRAM0ECCME	SRAM0 ECC multi-bits non-correction event
SYSCFG_INT_SRAM0ECCSE	SRAM0 ECC single bit correction event
SYSCFG_INT_FLASHHECC	Flash ECC NMI interrupt
SYSCFG_INT_CKMNMI	HXTAL clock moniator NMI interrupt
SYSCFG_INT_NMIPI_N	NMI pin interrupt
SYSCFG_INT_SRAM1ECCME	SRAM1 ECC multi-bits non-correction event
SYSCFG_INT_SRAM1ECCSE	SRAM1 ECC single bit correction event
SYSCFG_INT_TCMSTRAMECCME	TCMSRAM ECC multi-bits non-correction event
SYSCFG_INT_TCMSTRAMECCSE	TCMSRAM ECC single bit correction event

### Enum syscfg\_flag\_enum

**Table 3-1099. Enum syscfg\_flag\_enum**

enum name	Function description
SYSCFG_INT_FLAG_SRAM0ECCME	SRAM0 ECC multi-bits non-correction event flag
SYSCFG_INT_FLAG_SRAM0ECCSE	SRAM0 ECC single bit correction event flag
SYSCFG_INT_FLAG_FLASHHECC	Flash ECC NMI interrupt flag
SYSCFG_INT_FLAG_CKMNMI	HXTAL clock moniator NMI interrupt flag
SYSCFG_INT_FLAG_NMIPIN	NMI pin interrupt flag

enum name	Function description
SYSCFG_INT_FLAG_SRAM1ECCME	SRAM1 ECC multi-bits non-correction event flag
SYSCFG_INT_FLAG_SRAM1ECCSE	SRAM1 ECC single bit correction event flag
SYSCFG_INT_FLAG_TCMSRAM_ECCME	TCMSRAM ECC multi-bits non-correction event flag
SYSCFG_INT_FLAG_TCMSRAM_ECCSE	TCMSRAM ECC single bit correction event flag

### Enum syscfg\_sram\_serrbits\_enum

Table 3-1100. Enum syscfg\_sram\_serrbits\_enum

enum name	Function description
SYSCFG_SRAM0_SERRBITS	SRAM0 ECC single-bit correctable error bit
SYSCFG_SRAM1_SERRBITS	SRAM1 ECC single-bit correctable error bit
SYSCFG_TCMSRAM_SERRBITS	TCMSRAM ECC single-bit correctable error bit

### Enum syscfg\_sram\_erraddr\_enum

Table 3-1101. Enum syscfg\_sram\_erraddr\_enum

enum name	Function description
SYSCFG_SRAM0_ERR_ADDR	SRAM0 ECC error address
SYSCFG_SRAM1_ERR_ADDR	SRAM1 ECC error address
SYSCFG_TCMSRAM_ERR_ADDR	TCMSRAM ECC error address

### Enum timer\_channel\_input\_enum

Table 3-1102. Enum timer\_channel\_input\_enum

enum name	Function description
TIMER7_C10_INPUT_TIMER7_CH0	select TIMER7 CH0 output as TIMER7 C10
TIMER7_C10_INPUT_CMP0_OUT	select CMP0 output as TIMER7 C10
TIMER7_C10_INPUT_CMP1_OUT	select CMP1 output as TIMER7 C10
TIMER7_C10_INPUT_CMP2_OUT	select CMP2 output as TIMER7 C10



enum name	Function description
TIMER7_CIO_INPUT_C MP3_OUT	select CMP3 output as TIMER7 CIO
TIMER7_CIO_INPUT_C LA0_OUT	select CLA0 output as TIMER7 CIO
TIMER7_CI1_INPUT_T IMER7_CH1	select TIMER7 CH1 output as TIMER7 CI1
TIMER7_CI1_INPUT_C LA1_OUT	select CLA1 output as TIMER7 CI1
TIMER7_CI2_INPUT_T IMER7_CH2	select TIMER7 CH2 output as TIMER7 CI2
TIMER7_CI2_INPUT_C LA2_OUT	select CLA2 output as TIMER7 CI2
TIMER7_CI3_INPUT_T IMER7_CH3	select TIMER7 CH3 output as TIMER7 CI3
TIMER7_CI3_INPUT_C LA3_OUT	select CLA3 output as TIMER7 CI3
TIMER0_CIO_INPUT_T IMER0_CH0	select TIMER0 CH0 output as TIMER0 CIO
TIMER0_CIO_INPUT_C MP0_OUT	select CMP0 output as TIMER0 CIO
TIMER0_CIO_INPUT_C MP1_OUT	select CMP1 output as TIMER0 CIO
TIMER0_CIO_INPUT_C MP2_OUT	select CMP2 output as TIMER0 CIO
TIMER0_CIO_INPUT_C MP3_OUT	select CMP3 output as TIMER0 CIO
TIMER0_CIO_INPUT_C LA0_OUT	select CLA0 output as TIMER0 CIO
TIMER0_CI1_INPUT_T IMER0_CH1	select TIMER0 CH1 as TIMER0 CI1
TIMER0_CI1_INPUT_C LA1_OUT	select CLA1 output as TIMER0 CI1
TIMER0_CI2_INPUT_T IMER0_CH2	select TIMER0 CH2 as TIMER0 CI2
TIMER0_CI2_INPUT_C LA2_OUT	select CLA2 output as TIMER0 CI2
TIMER0_CI3_INPUT_T IMER0_CH3	select TIMER0 CH3 as TIMER0 CI3
TIMER0_CI3_INPUT_C LA3_OUT	select CLA3 output as TIMER0 CI3
TIMER19_CIO_INPUT_ TIMER19_CH0	select TIMER190 CH0 output as TIMER19 CIO

enum name	Function description
TIMER19_CIO_INPUT_CMP0_OUT	select CMP0 output as TIMER19 CIO
TIMER19_CIO_INPUT_CMP1_OUT	select CMP1 output as TIMER19 CIO
TIMER19_CIO_INPUT_CMP2_OUT	select CMP2 output as TIMER19 CIO
TIMER19_CIO_INPUT_CMP3_OUT	select CMP3 output as TIMER19 CIO
TIMER19_CIO_INPUT_CLA0_OUT	select CLA0 output as TIMER19 CIO
TIMER19_C11_INPUT_TIMER19_CH1	select TIMER19 CH1 as TIMER19 C11
TIMER19_C11_INPUT_CLA1_OUT	select CLA1 output as TIMER19 C11
TIMER19_C12_INPUT_TIMER19_CH2	select TIMER19 CH2 as TIMER19 C12
TIMER19_C12_INPUT_CLA2_OUT	select CLA2 output as TIMER19 C12
TIMER19_C13_INPUT_TIMER19_CH3	select TIMER19 CH3 as TIMER19 C13
TIMER19_C13_INPUT_CLA3_OUT	select CLA3 output as TIMER19 C13
TIMER2_CIO_INPUT_TIMER2_CH0	select TIMER2 CH0 as TIMER2 CIO
TIMER2_CIO_INPUT_CMP0_OUT	select CMP0 output as TIMER2 CIO
TIMER2_CIO_INPUT_CMP1_OUT	select CMP1 output as TIMER2 CIO
TIMER2_CIO_INPUT_CMP2_OUT	select CMP2 output as TIMER2 CIO
TIMER2_CIO_INPUT_CMP3_OUT	select CMP3 output as TIMER2 CIO
TIMER2_CIO_INPUT_CMP4_OUT	select CMP4 output as TIMER2 CIO
TIMER2_CIO_INPUT_CMP5_OUT	select CMP5 output as TIMER2 CIO
TIMER2_CIO_INPUT_CMP6_OUT	select CMP6 output as TIMER2 CIO
TIMER2_CIO_INPUT_CLA0_OUT	select CLA0 output as TIMER19 CIO
TIMER2_C11_INPUT_TIMER2_CH1	select TIMER2 CH1 as TIMER2 C11

enum name	Function description
TIMER2_CI1_INPUT_C MP0_OUT	select CMP0 output as TIMER2 CI1
TIMER2_CI1_INPUT_C MP1_OUT	select CMP1 output as TIMER2 CI1
TIMER2_CI1_INPUT_C MP2_OUT	select CMP2 output as TIMER2 CI1
TIMER2_CI1_INPUT_C MP3_OUT	select CMP3 output as TIMER2 CI1
TIMER2_CI1_INPUT_C MP4_OUT	select CMP4 output as TIMER2 CI1
TIMER2_CI1_INPUT_C MP5_OUT	select CMP5 output as TIMER2 CI1
TIMER2_CI1_INPUT_C MP6_OUT	select CMP6 output as TIMER2 CI1
TIMER2_CI1_INPUT_C LA1_OUT	select CLA1 output as TIMER2 CI1
TIMER2_CI2_INPUT_T IMER2_CH2	select TIMER2 CH2 as TIMER2 CI2
TIMER2_CI2_INPUT_C MP2_OUT	select CMP2 output as TIMER2 CI2
TIMER2_CI2_INPUT_C LA2_OUT	select CLA2 output as TIMER2 CI2
TIMER2_CI3_INPUT_T IMER2_CH3	select TIMER2 CH3 as TIMER2 CI3
TIMER2_CI3_INPUT_C LA3_OUT	select CLA3 output as TIMER2 CI3
TIMER1_CI0_INPUT_T IMER1_CH0	select TIMER1 CH0 as TIMER1 CI0
TIMER1_CI0_INPUT_C MP0_OUT	select CMP0 output as TIMER1 CI0
TIMER1_CI0_INPUT_C MP1_OUT	select CMP1 output as TIMER1 CI0
TIMER1_CI0_INPUT_C MP2_OUT	select CMP2 output as TIMER1 CI0
TIMER1_CI0_INPUT_C MP3_OUT	select CMP3 output as TIMER1 CI0
TIMER1_CI0_INPUT_C MP4_OUT	select CMP4 output as TIMER1 CI0
TIMER1_CI0_INPUT_C LA0_OUT	select CLA0 output as TIMER1 CI0
TIMER1_CI1_INPUT_T IMER1_CH1	select TIMER1 CH1 as TIMER1 CI1

enum name	Function description
TIMER1_CI1_INPUT_C MP0_OUT	select CMP0 output as TIMER1 CI1
TIMER1_CI1_INPUT_C MP1_OUT	select CMP1 output as TIMER1 CI1
TIMER1_CI1_INPUT_C MP2_OUT	select CMP2 output as TIMER1 CI1
TIMER1_CI1_INPUT_C MP3_OUT	select CMP3 output as TIMER1 CI1
TIMER1_CI1_INPUT_C MP5_OUT	select CMP4 output as TIMER1 CI1
TIMER1_CI1_INPUT_C LA1_OUT	select CLA1 output as TIMER1 CI1
TIMER1_CI2_INPUT_T IMER1_CH2	select TIMER1 CH2 as TIMER1 CI2
TIMER1_CI2_INPUT_C MP3_OUT	select CMP3 output as TIMER1 CI2
TIMER1_CI2_INPUT_C LA2_OUT	select CLA2 output as TIMER1 CI2
TIMER1_CI3_INPUT_T IMER1_CH3	select TIMER1 CH3 as TIMER1 CI3
TIMER1_CI3_INPUT_C MP0_OUT	select CMP0 output as TIMER1 CI3
TIMER1_CI3_INPUT_C MP1_OUT	select CMP1 output as TIMER1 CI3
TIMER1_CI3_INPUT_C MP1_OUT	select CMP2 output as TIMER1 CI3
TIMER1_CI3_INPUT_C LA3_OUT	select CLA3 output as TIMER1 CI3
TIMER4_CIO_INPUT_T IMER4_CH0	select TIMER4 CH0 as TIMER4 CIO
TIMER4_CIO_INPUT_I RC32K	select IRC32K as TIMER4 CIO
TIMER4_CIO_INPUT_ LXTAL	select LXTAL output as TIMER4 CIO
TIMER4_CIO_INPUT_R TC_WAKEUP	select RTC_WAKEUP output as TIMER4 CIO
TIMER4_CIO_INPUT_C MP0_OUT	select CMP0 output as TIMER4 CIO
TIMER4_CIO_INPUT_C MP1_OUT	select CMP1 output as TIMER4 CIO
TIMER4_CIO_INPUT_C MP2_OUT	select CMP2 output as TIMER4 CIO

enum name	Function description
TIMER4_CIO_INPUT_C MP3_OUT	select CMP3 output as TIMER4 CI0
TIMER4_CIO_INPUT_C MP4_OUT	select CMP4 output as TIMER4 CI0
TIMER4_CIO_INPUT_C MP5_OUT	select CMP5 output as TIMER4 CI0
TIMER4_CIO_INPUT_C MP6_OUT	select CMP6 output as TIMER4 CI0
TIMER4_CIO_INPUT_C LA0_OUT	select CLA0 output as TIMER4 CI0
TIMER4_CI1_INPUT_T IMER4_CH1	select TIMER4 CH1 as TIMER4 CI1
TIMER4_CI1_INPUT_C MP0_OUT	select CMP0 output as TIMER4 CI1
TIMER4_CI1_INPUT_C MP1_OUT	select CMP1 output as TIMER4 CI1
TIMER4_CI1_INPUT_C MP2_OUT	select CMP2 output as TIMER4 CI1
TIMER4_CI1_INPUT_C MP3_OUT	select CMP3 output as TIMER4 CI1
TIMER4_CI1_INPUT_C MP4_OUT	select CMP4 output as TIMER4 CI1
TIMER4_CI1_INPUT_C MP5_OUT	select CMP5 output as TIMER4 CI1
TIMER4_CI1_INPUT_C MP6_OUT	select CMP6 output as TIMER4 CI1
TIMER4_CI1_INPUT_C LA1_OUT	select CLA1 output as TIMER4 CI1
TIMER4_CI2_INPUT_T IMER4_CH2	select TIMER4 CH2 as TIMER4 CI2
TIMER4_CI2_INPUT_C LA2_OUT	select CLA2 output as TIMER4 CI2
TIMER4_CI3_INPUT_T IMER4_CH3	select TIMER4 CH3 as TIMER4 CI3
TIMER4_CI3_INPUT_C LA3_OUT	select CLA3 output as TIMER4 CI3
TIMER3_CIO_INPUT_T IMER3_CH0	select TIMER3 CH0 as TIMER3 CI0
TIMER3_CIO_INPUT_C MP0_OUT	select CMP0 output as TIMER3 CI0
TIMER3_CIO_INPUT_C MP1_OUT	select CMP1 output as TIMER3 CI0

enum name	Function description
TIMER3_C10_INPUT_C MP2_OUT	select CMP2 output as TIMER3 CI0
TIMER3_C10_INPUT_C MP3_OUT	select CMP3 output as TIMER3 CI0
TIMER3_C10_INPUT_C MP4_OUT	select CMP4 output as TIMER3 CI0
TIMER3_C10_INPUT_C MP5_OUT	select CMP5 output as TIMER3 CI0
TIMER3_C10_INPUT_C MP6_OUT	select CMP6 output as TIMER3 CI0
TIMER3_C10_INPUT_C LA0_OUT	select CLA0 output as TIMER3 CI0
TIMER3_C11_INPUT_T IMER3_CH1	select TIMER3 CH1 as TIMER3 CI1
TIMER3_C11_INPUT_C MP0_OUT	select CMP0 output as TIMER3 CI1
TIMER3_C11_INPUT_C MP1_OUT	select CMP1 output as TIMER3 CI1
TIMER3_C11_INPUT_C MP2_OUT	select CMP2 output as TIMER3 CI1
TIMER3_C11_INPUT_C MP3_OUT	select CMP3 output as TIMER3 CI1
TIMER3_C11_INPUT_C MP4_OUT	select CMP4 output as TIMER3 CI1
TIMER3_C11_INPUT_C MP5_OUT	select CMP5 output as TIMER3 CI1
TIMER3_C11_INPUT_C MP6_OUT	select CMP6 output as TIMER3 CI1
TIMER3_C11_INPUT_C LA1_OUT	select CLA1 output as TIMER3 CI1
TIMER3_C12_INPUT_T IMER3_CH2	select TIMER3 CH2 as TIMER3 CI2
TIMER3_C12_INPUT_C MP4_OUT	select CMP4 output as TIMER3 CI2
TIMER3_C12_INPUT_C LA2_OUT	select CLA2 output as TIMER3 CI2
TIMER3_C13_INPUT_T IMER3_CH3	select TIMER3 CH3 as TIMER3 CI3
TIMER3_C13_INPUT_C MP5_OUT	select CMP5output as TIMER3 CI3
TIMER3_C12_INPUT_C LA3_OUT	select CLA3 output as TIMER3 CI3

enum name	Function description
TIMER14_CIO_INPUT_ TIMER14_CH0	select TIMER14 CH0 as TIMER14 CIO
TIMER14_CIO_INPUT_ LXTAL	select LXTAL as TIMER14 CIO
TIMER14_CIO_INPUT_ CMP0_OUT	select CMP0 output as TIMER3 CIO
TIMER14_CIO_INPUT_ CMP1_OUT	select CMP1 output as TIMER14 CIO
TIMER14_CIO_INPUT_ CMP4_OUT	select CMP4 output as TIMER14 CIO
TIMER14_CIO_INPUT_ CMP6_OUT	select CMP6 output as TIMER14 CIO
TIMER14_CIO_INPUT_ CLA0_OUT	select CLA0 output as TIMER14 CIO
TIMER14_CIO_INPUT_ CLA1_OUT	select CLA1 output as TIMER14 CIO
TIMER14_CIO_INPUT_ CLA2_OUT	select CLA2 output as TIMER14 CIO
TIMER14_CIO_INPUT_ CLA3_OUT	select CLA3 output as TIMER14 CIO
TIMER14_CI1_INPUT_ TIMER14_CH1	select TIMER14 CH1 as TIMER14 CI1
TIMER14_CI1_INPUT_ CMP1_OUT	select CMP1 output as TIMER14 CI1
TIMER14_CI1_INPUT_ CMP2_OUT	select CMP2 output as TIMER14 CI1
TIMER14_CI1_INPUT_ CMP5_OUT	select CMP5 output as TIMER14 CI1
TIMER14_CI1_INPUT_ CMP6_OUT	select CMP6 output as TIMER14 CI1
TIMER15_CIO_INPUT_ TIMER15_CH0	select TIMER15 CH0 as TIMER15 CIO
TIMER15_CIO_INPUT_ CMP5_OUT	select CMP5 output as TIMER15 CIO
TIMER15_CIO_INPUT_ CKOUT	select CKOUT as TIMER15 CIO
TIMER15_CIO_INPUT_ HXTAL_DIV32	select HXTAL/32 as TIMER15 CIO
TIMER15_CIO_INPUT_ RTC_CLOCK	select RTC_CLOCK as TIMER15 CIO
TIMER15_CIO_INPUT_ LXTAL	select LXTAL as TIMER15 CIO

enum name	Function description
TIMER15_CIO_INPUT_I RC32K	select IRC32K as TIMER15 CIO
TIMER15_CIO_INPUT_ CLA0_OUT	select CLA0 output as TIMER15 CIO
TIMER15_CIO_INPUT_ CLA1_OUT	select CLA1 output as TIMER15 CIO
TIMER15_CIO_INPUT_ CLA2_OUT	select CLA2 output as TIMER15 CIO
TIMER15_CIO_INPUT_ CLA3_OUT	select CLA3 output as TIMER15 CIO
TIMER15_CIO_INPUT_ HXTAL	select HXTAL as TIMER15 CIO
TIMER15_CIO_INPUT_ WKUP_IT	select WKUP_IT as TIMER15 CIO
TIMER16_CIO_INPUT_ TIMER16_CH0	select TIMER16 CH0 as TIMER16 CIO
TIMER16_CIO_INPUT_ CMP4_OUT	select CMP4 output as TIMER16 CIO
TIMER16_CIO_INPUT_ CKOUT	select CKOUT as TIMER16 CIO
TIMER16_CIO_INPUT_ HXTAL_DIV32	select HXTAL as TIMER16 CIO
TIMER16_CIO_INPUT_ RTC_CLOCK	select RTC_CLOCK as TIMER16 CIO
TIMER16_CIO_INPUT_ LXTAL	select LXTAL as TIMER16 CIO
TIMER16_CIO_INPUT_I RC32K	select IRC32K as TIMER16 CIO
TIMER16_CIO_INPU T_CLA0_OUT	select CLA0 output as TIMER16 CIO
TIMER16_CIO_INPU T_CLA1_OUT	select CLA1 output as TIMER16 CIO
TIMER16_CIO_INPU T_CLA2_OUT	select CLA2 output as TIMER16 CIO
TIMER16_CIO_INPU T_CLA3_OUT	select CLA3 output as TIMER16 CIO
TIMER16_CIO_INPUT_ HXTAL	select HXTAL as TIMER16 CIO

### syscfg\_deinit

The description of syscfg\_deinit is shown as below:



Table 3-1103. Function syscfg\_deinit

Function name	syscfg_deinit
Function prototype	void syscfg_deinit(void);
Function descriptions	reset the SYSCFG registers
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset SYSCFG registers */
```

```
syscfg_deinit();
```

### syscfg\_i2c\_fast\_mode\_plus\_enable

The description of syscfg\_i2c\_fast\_mode\_plus\_enable is shown as below:

Table 3-1104. Function syscfg\_i2c\_fast\_mode\_plus\_enable

Function name	syscfg_i2c_fast_mode_plus_enable
Function prototype	void syscfg_i2c_fast_mode_plus_enable(uint32_t i2c_fmp);
Function descriptions	enable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
Precondition	-
The called functions	-
Input parameter{in}	
i2c_fmp	I2C fast mode plus function
SYSCFG_I2C0_FMP	I2C0 fast mode plus
SYSCFG_I2C1_FMP	I2C1 fast mode plus
SYSCFG_I2C2_FMP	I2C2 fast mode plus
SYSCFG_I2C3_FMP	I2C3 fast mode plus
SYSCFG_I2C_FMP_P B6	I2C fast mode plus on PB6 pin
SYSCFG_I2C_FMP_P B7	I2C fast mode plus on PB7 pin
SYSCFG_I2C_FMP_P B8	I2C fast mode plus on PB8 pin
SYSCFG_I2C_FMP_P B9	I2C fast mode plus on PB9 pin
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* enable I2C0 fast mode plus function */
```

```
syscfg_i2c_fast_mode_plus_enable(SYSCFG_I2C0_FMP);
```

### syscfg\_i2c\_fast\_mode\_plus\_disable

The description of syscfg\_i2c\_fast\_mode\_plus\_disable is shown as below:

**Table 3-1105. Function syscfg\_i2c\_fast\_mode\_plus\_disable**

Function name	syscfg_i2c_fast_mode_plus_disable
Function prototype	void syscfg_i2c_fast_mode_plus_disable(uint32_t i2c_fmp);
Function descriptions	disable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
Precondition	-
The called functions	-
Input parameter{in}	
i2c_fmp	I2C fast mode plus function
SYSCFG_I2C0_FMP	I2C0 fast mode plus
SYSCFG_I2C1_FMP	I2C1 fast mode plus
SYSCFG_I2C2_FMP	I2C2 fast mode plus
SYSCFG_I2C3_FMP	I2C3 fast mode plus
SYSCFG_I2C_FMP_P B6	I2C fast mode plus on PB6 pin
SYSCFG_I2C_FMP_P B7	I2C fast mode plus on PB7 pin
SYSCFG_I2C_FMP_P B8	I2C fast mode plus on PB8 pin
SYSCFG_I2C_FMP_P B9	I2C fast mode plus on PB9 pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C0 fast mode plus function */
```

```
syscfg_i2c_fast_mode_plus_disable(SYSCFG_I2C0_FMP);
```

## syscfg\_exti\_line\_config

The description of syscfg\_exti\_line\_config is shown as below:

**Table 3-1106. Function syscfg\_exti\_line\_config**

<b>Function name</b>	syscfg_exti_line_config
<b>Function prototype</b>	void syscfg_exti_line_config(uint8_t exti_port, uint8_t exti_pin);
<b>Function descriptions</b>	configure the GPIO pin as EXTI Line
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exti_port</b>	specify the GPIO port used in EXTI
<i>EXTI_SOURCE_GPIOx</i>	x = A,B,C,D,E,F,G
<b>Input parameter{in}</b>	
<b>exti_pin</b>	specify the EXTI line
<i>EXTI_SOURCE_PINx</i>	GPIOA x = 0..15, GPIOB x = 0..15, GPIOC x = 0..15, GPIOD x = 0..15, GPIOE x = 0..15, GPIOF x = 0..15, GPIOG x = 0..15
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the PA0 pin as EXTI Line */
```

```
syscfg_exti_line_config(EXTI_SOURCE_GPIOA, EXTI_SOURCE_PIN0);
```

## syscfg\_pin\_reset\_mode\_config

The description of syscfg\_pin\_reset\_mode\_config is shown as below:

**Table 3-1107. Function syscfg\_pin\_reset\_mode\_config**

<b>Function name</b>	syscfg_pin_reset_mode_config
<b>Function prototype</b>	void syscfg_pin_reset_mode_config(uint32_t syscfg_pin_reset_mode);
<b>Function descriptions</b>	configure GPIO reset mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>syscfg_pin_reset_mode</b>	configure GPIO reset mode
<i>SYSCFG_PIN_NRST</i>	GPIO pin configuration will retain state across any reset event except for the POR event
<i>SYSCFG_PIN_RST</i>	GPIO pin configuration is reset when any reset event occurs
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* GPIO remain state after reset */
```

```
syscfg_pin_reset_mode_config(SYS_CFG_PIN_NRST);
```

### syscfg\_trigsel\_cla\_reset\_mode\_config

The description of syscfg\_trigsel\_cla\_reset\_mode\_config is shown as below:

**Table 3-1108. Function syscfg\_trigsel\_cla\_reset\_mode\_config**

Function name	syscfg_trigsel_cla_reset_mode_config
Function prototype	void syscfg_trigsel_cla_reset_mode_config(uint32_t syscfg_trigsel_reset_mode);
Function descriptions	configure Trigsel cla reset mode
Precondition	-
The called functions	-
Input parameter{in}	
syscfg_trigsel_reset_mode	configure Trigsel cla reset mode
SYS_CFG_TRGSEL_CLA_NRST	Trigsel CLA configuration will retain state across any reset event except for the POR event
SYS_CFG_TRGSEL_CLA_RST	Trigsel CLA configuration is reset when any reset event occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* Trigsel CLA remain state after reset */
```

```
syscfg_trigsel_cla_reset_mode_config(SYS_CFG_TRGSEL_CLA_NRST);
```

### syscfg\_lockup\_enable

The description of syscfg\_lockup\_enable is shown as below:

**Table 3-1109. Function syscfg\_lockup\_enable**

Function name	syscfg_lockup_enable
Function prototype	void syscfg_lockup_enable(uint32_t lockup);
Function descriptions	configure module lockup
Precondition	-
The called functions	-

Input parameter{in}	
<b>lockup</b>	lockup configuration
<i>SYSCFG_LVD_LOCKUP</i>	LVD signal
<i>SYSCFG_LOCKUP_LOCKUP</i>	CPU lockup signal
<i>SYSCFG_FLASH_LOCKUP</i>	Flash ECC double error signal
<i>SYSCFG_SRAM1_LOCKUP</i>	SRAM1 ECC double error signal
<i>SYSCFG_SRAM0_LOCKUP</i>	SRAM0 ECC double error signal
<i>SYSCFG_TCMSRAM_LOCKUP</i>	TCMSRAM ECC double error signal
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable module lockup function */
```

```
syscfg_lockup_enable(SYSCFG_CFG2_LOCKUP_LOCK);
```

### syscfg\_timer\_input\_source\_select

The description of syscfg\_timer\_input\_source\_select is shown as below:

**Table 3-1110. Function syscfg\_timer\_input\_source\_select**

<b>Function name</b>	syscfg_timer_input_source_select
<b>Function prototype</b>	void syscfg_timer_input_source_select(timer_channel_input_enum timer_input);
<b>Function descriptions</b>	select timer channel input source
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_input</b>	TIMER channel input select, refer to <a href="#">Table 3-1098. Enum syscfg_interrupt_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select timer channel input source */
```

```
syscfg_timer_input_source_select(TIMER7_CIO_INPUT_TIMER7_CH0);
```

## syscfg\_flash\_bank\_remap\_set

The description of syscfg\_flash\_bank\_remap\_set is shown as below:

**Table 3-1111. Function syscfg\_flash\_bank\_remap\_set**

Function name	syscfg_flash_bank_remap_set
Function prototype	void syscfg_flash_bank_remap_set(uint32_t value);
Function descriptions	switch flash Bank0 and Bank1 addrs
Precondition	-
The called functions	-
Input parameter{in}	
value	switch flash Bank0 and Bank1 addrs
SYSCFG_FLASH_BANK0_MAPPED	Flash Bank 1 mapped at 0x08000000,and Flash Bank 0 mapped at 0x08040000
SYSCFG_FLASH_BANK1_MAPPED	Flash Bank 0 mapped at 0x08000000,and Flash Bank 1 mapped at 0x08040000
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* Flash Bank 1 mapped at 0x08000000,and Flash Bank 0 mapped at 0x08040000 */
```

```
syscfg_flash_bank_remap_set(SYSCFG_FLASH_BANK0_MAPPED);
```

## syscfg\_bootmode\_memmap\_select

The description of syscfg\_bootmode\_memmap\_select is shown as below:

**Table 3-1112. Function syscfg\_bootmode\_memmap\_select**

Function name	syscfg_bootmode_memmap_select
Function prototype	void syscfg_bootmode_memmap_select(uint32_t mem_select);
Function descriptions	select memory mapping
Precondition	-
The called functions	-
Input parameter{in}	
mem_select	mem_select
SYSCFG_MAIN_FLASH_MEMORY	Main Flash memory mapped at 0x00000000
SYSCFG_SYSTEM_FLASH_MEMORY	System Flash memory mapped at 0x00000000

<code>SYSCFG_EXMC_MEMORY</code>	EXMC memory mapped at 0x00000000
<code>SYSCFG_SRAM0_MEMORY</code>	SRAM0 mapped at 0x00000000
<code>SYSCFG_QSPI_MEMORY</code>	QSPI memory mapped at 0x00000000
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* Main Flash memory mapped at 0x00000000 */
syscfg_bootmode_memmap_select(SYSCFG_MAIN_FLASH_MEMORY);
```

### syscfg\_sram\_ecc\_single\_correctable\_bit\_get

The description of `syscfg_sram_ecc_single_correctable_bit_get` is shown as below:

**Table 3-1113. Function `syscfg_sram_ecc_single_correctable_bit_get`**

<b>Function name</b>	<code>syscfg_sram_ecc_single_correctable_bit_get</code>
<b>Function prototype</b>	uint32_t <code>syscfg_sram_ecc_single_correctable_bit_get(syscfg_sram_serrbits_enum sram);</code>
<b>Function descriptions</b>	get SRAM ECC error bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sram</b>	SRAM ECC error bits, refer to <a href="#">Table 3-1100. Enum <code>syscfg_sram_serrbits_enum</code></a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>error_bits</b>	SRAM ECC error bits

Example:

```
/* get SRAM0 SRAM0 ECC single-bit correctable error bits */
syscfg_sram_ecc_single_correctable_bit_get(SYSCFG_SRAM0_SERRBITS);
```

## syscfg\_sram\_ecc\_error\_address\_get

The description of syscfg\_sram\_ecc\_error\_address\_get is shown as below:

**Table 3-1114. Function syscfg\_sram\_ecc\_error\_address\_get**

<b>Function name</b>	syscfg_sram_ecc_error_address_get
<b>Function prototype</b>	uint32_t syscfg_sram_ecc_error_address_get(syscfg_sram_erraddr_enum sram);
<b>Function descriptions</b>	get SRAM ECC error address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sram</b>	SRAM ECC error address, refer to <a href="#">Table 3-1101. Enum syscfg_sram_erraddr_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>addr</b>	error address

Example:

```
/* get SRAM0 ECC error address */
```

```
syscfg_sram_ecc_error_address_get(SYS_CFG_SRAM0_ERR_ADDR);
```

## syscfg\_tcmsram\_erase

The description of syscfg\_tcmsram\_erase is shown as below:

**Table 3-1115. Function syscfg\_tcmsram\_erase**

<b>Function name</b>	syscfg_tcmsram_erase
<b>Function prototype</b>	void syscfg_tcmsram_erase(void);
<b>Function descriptions</b>	erase TCMSRAM
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* erase TCMSRAM */
```

```
syscfg_tcmsram_erase ();
```



## syscfg\_tcmsram\_erase\_lock

The description of syscfg\_tcmsram\_erase\_lock is shown as below:

**Table 3-1116. Function syscfg\_tcmsram\_erase\_lock**

<b>Function name</b>	syscfg_tcmsram_erase_lock
<b>Function prototype</b>	void syscfg_tcmsram_erase_lock(void);
<b>Function descriptions</b>	lock the TCMSRAM erase
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* lock the TCMSRAM erase */
syscfg_tcmsram_erase_lock();
```

## syscfg\_tcmsram\_erase\_unlock

The description of syscfg\_tcmsram\_erase\_unlock is shown as below:

**Table 3-1117. Function syscfg\_tcmsram\_erase\_unlock**

<b>Function name</b>	syscfg_tcmsram_erase_unlock
<b>Function prototype</b>	void syscfg_tcmsram_erase_unlock (void);
<b>Function descriptions</b>	unlock the TCMSRAM erase
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* unlock the TCMSRAM erase */
syscfg_tcmsram_erase_unlock();
```

## syscfg\_tcmsram\_page\_wp\_enable

The description of syscfg\_tcmsram\_page\_wp\_enable is shown as below:

**Table 3-1118. Function syscfg\_tcmsram\_page\_wp\_enable**

<b>Function name</b>	syscfg_tcmsram_page_wp_enable
<b>Function prototype</b>	void syscfg_tcmsram_page_wp_enable(uint32_t pagex);
<b>Function descriptions</b>	enable TCMSRAM page x write protection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pagex</b>	TCMSRAM page x write protection enable,
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TCMSRAM page 0 write protection */
syscfg_tcmsram_page_wp_enable(SYSCFG_TCMSRAMWP_P0WPEN);
```

## syscfg\_io\_compensation\_config

The description of syscfg\_io\_compensation\_config is shown as below:

**Table 3-1119. Function syscfg\_io\_compensation\_config**

<b>Function name</b>	syscfg_io_compensation_config
<b>Function prototype</b>	void syscfg_io_compensation_config(void);
<b>Function descriptions</b>	configure I/O compensation cell
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>syscfg_cps</b>	I/O compensation cell config
<b>SYSCFG_IO_COMPENSATION_ENABLE</b>	I/O compensation cell enable
<b>SYSCFG_IO_COMPENSATION_DISABLE</b>	I/O compensation cell disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* I/O compensation cell enable */
```

```
syscfg_io_compensation_config(SYSCFG_IO_COMPENSATION_ENABLE);
```

### syscfg\_fpu\_interrupt\_enable

The description of syscfg\_fpu\_interrupt\_enable is shown as below:

**Table 3-1120. Function syscfg\_fpu\_interrupt\_enable**

<b>Function name</b>	syscfg_fpu_interrupt_enable
<b>Function prototype</b>	void syscfg_fpu_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable FPU interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
interrupt	FPU interrupt
SYSCFG_FPUINT_INV ALID_OPERATION	invalid operation interrupt
SYSCFG_FPUINT_DIV 0	divide-by-zero interrupt
SYSCFG_FPUINT_OV ERFLOW	overflow interrupt
SYSCFG_FPUINT_UN DERFLOW	underflow interrupt
SYSCFG_FPUINT_INP UT_ABNORMAL	input abnormal interrupt
SYSCFG_FPUINT_INE XACT	inexact interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FPU inexact interrupt */
syscfg_fpu_interrupt_enable(SYSCFG_FPUINT_INEXACT);
```

### syscfg\_fpu\_interrupt\_disable

The description of syscfg\_fpu\_interrupt\_disable is shown as below:

**Table 3-1121. Function syscfg\_fpu\_interrupt\_disable**

<b>Function name</b>	syscfg_fpu_interrupt_disable
<b>Function prototype</b>	void syscfg_fpu_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable FPU interrupt
<b>Precondition</b>	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FPU interrupt
<i>SYSCFG_FPUINT_INV ALID_OPERATION</i>	invalid operation interrupt
<i>SYSCFG_FPUINT_DIV 0</i>	divide-by-zero interrupt
<i>SYSCFG_FPUINT_OV ERFLOW</i>	overflow interrupt
<i>SYSCFG_FPUINT_UN DERFLOW</i>	underflow interrupt
<i>SYSCFG_FPUINT_INP UT_ABNORMAL</i>	input abnormal interrupt
<i>SYSCFG_FPUINT_INE XACT</i>	inexact interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FPU inexact interrupt */
syscfg_fpu_interrupt_disable(SYSCFG_FPUINT_INEXACT);
```

### syscfg\_interrupt\_flag\_get

The description of syscfg\_interrupt\_flag\_get is shown as below:

**Table 3-1122. Function syscfg\_interrupt\_flag\_get**

<b>Function name</b>	syscfg_interrupt_flag_get
<b>Function prototype</b>	FlagStatus syscfg_interrupt_flag_get(syscfg_flag_enum int_flag);
<b>Function descriptions</b>	get interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	Get interrupt flag, refer to <a href="#">Table 3-1099. Enum syscfg_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	Interrupt flag status
<i>SET</i>	Flag set
<i>RESET</i>	Flag is not set

Example:

```
/* get the SRAM0 ECC multi-bits non-correction event flag */
```

```
FlagStatus Flag = RESET;
```

```
Flag = syscfg_interrupt_flag_get(SYSCFG_INT_FLAG_SRAM0ECCME);
```

### syscfg\_interrupt\_flag\_clear

The description of syscfg\_interrupt\_flag\_clear is shown as below:

**Table 3-1123. Function syscfg\_interrupt\_flag\_clear**

<b>Function name</b>	syscfg_interrupt_flag_clear
<b>Function prototype</b>	void syscfg_interrupt_flag_clear(syscfg_flag_enum int_flag);
<b>Function descriptions</b>	clear interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	Clear interrupt flag, refer to <a href="#">Table 3-1099. Enum syscfg_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the SRAM0 ECC multi-bits non-correction event flag */
```

```
syscfg_interrupt_flag_clear (SYSCFG_INT_FLAG_SRAM0ECCME);
```

### syscfg\_interrupt\_enable

The description of syscfg\_interrupt\_enable is shown as below:

**Table 3-1124. Function syscfg\_interrupt\_enable**

<b>Function name</b>	syscfg_interrupt_enable
<b>Function prototype</b>	void syscfg_interrupt_enable(syscfg_interrupt_enum interrupt);
<b>Function descriptions</b>	enable interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	Enable interrupt, refer to <a href="#">Table 3-1098. Enum syscfg_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the SRAM0 ECC multi-bits non-correction event flag */
```

```
syscfg_interrupt_enable(SYSCFG_INT_SRAM0ECCME);
```

### syscfg\_interrupt\_disable

The description of syscfg\_interrupt\_disable is shown as below:

**Table 3-1125. Function syscfg\_interrupt\_disable**

<b>Function name</b>	syscfg_interrupt_disable
<b>Function prototype</b>	void syscfg_interrupt_disable(syscfg_interrupt_enum interrupt);
<b>Function descriptions</b>	disable interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	Disable interrupt, refer to <a href="#">Table 3-1098. Enum syscfg_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* Disable the SRAM0 ECC multi-bits non-correction event flag */
```

```
syscfg_interrupt_disable(SYSCFG_INT_SRAM0ECCME);
```

### syscfg\_tcmsram\_busy\_flag\_get

The description of syscfg\_tcmsram\_busy\_flag\_get is shown as below:

**Table 3-1126. Function syscfg\_tcmsram\_busy\_flag\_get**

<b>Function name</b>	syscfg_tcmsram_busy_flag_get
<b>Function prototype</b>	FlagStatus syscfg_tcmsram_busy_flag_get(void);
<b>Function descriptions</b>	get tcmsram erase busy flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get tcmsram erase busy flag */
```

```
FlagStatus Flag = RESET;
Flag = syscfg_tcmsram_busy_flag_get();
```

### syscfg\_compensation\_cell\_ready\_flag\_get

The description of syscfg\_compensation\_cell\_ready\_flag\_get is shown as below:

**Table 3-1127. Function syscfg\_compensation\_cell\_ready\_flag\_get**

<b>Function name</b>	syscfg_compensation_cell_ready_flag_get
<b>Function prototype</b>	FlagStatus syscfg_compensation_cell_ready_flag_get(void);
<b>Function descriptions</b>	get compensation cell ready flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get compensation cell ready flag */

FlagStatus Flag = RESET;
Flag = syscfg_compensation_cell_ready_flag_get();
```

## 3.30. TIMER

The timers have a counter that can be used as an unsigned counter and supports both input capture and output compare. Timers (TIMERx) are divided into five sorts: advanced timer (TIMERx, x=0, 7, 19), general level0 timer (TIMERx, x=1~4), general level3 timer (TIMERx, x=14), general level4 timer (TIMERx, x=15, 16) and basic timer (TIMERx, x=5, 6). The specific functions of different types of timer are different. The TIMER registers are listed in chapter [3.30.1](#), the TIMER firmware functions are introduced in chapter [3.30.2](#).

### 3.30.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

**Table 3-1128. TIMERx Registers**

Registers	Descriptions
TIMER_CTL0	Control register 0
TIMER_CTL1	Control register 1
TIMER_SMCFG	Slave mode configuration register

Registers	Descriptions
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_CREP0	Counter repetition register 0
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_CCHP0	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_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
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_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_AFCTL0	TIMER alternate function control register 0
TIMER_AFCTL1	TIMER alternate function control register 1
TIMER_WDGP	TIMER watchdog counter period register
TIMER_CREP1	TIMER counter repetition register 1
TIMER_CCHP1	TIMER complementary channel protection register 1
TIMER_DECCTL	TIMER decoder control register



Registers	Descriptions
TIMER_CINITCTL	TIMER counter initial control register
TIMER_CINITV	TIMER counter initial value register
TIMER_DMACFG	DMA configuration register
TIMER_DMATB	DMA transfer buffer register
TIMER_CFG	Configuration register

### 3.30.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

**Table 3-1129. 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_adjustment_mode_config	configure the TIMER adjustment mode function
timer_prescaler_config	configure TIMER prescaler
timer_repetition_value_config	configure TIMER repetition register value
timer_runtime_repetition_value_read	configure TIMER runtime repetition register value
timer_autoreload_value_config	configure TIMER autoreload register value
timer_auto_reload_fract_value_config	configure TIMER fractional part of auto reload register value
timer_autoreload_value_read	read TIMER autoreload register value
timer_auto_reload_fract_value_read	read TIMER fractional part of auto reload register value
timer_counter_value_config	configure TIMER counter register value
timer_counter_read	read TIMER counter value
timer_prescaler_read	read TIMER prescaler value
timer_single_pulse_mode_config	configure TIMER single pulse mode
timer_delayable_single_pulse_mode_config	configure timer delayable single pulse mode
timer_update_source_config	configure TIMER update source
timer_ocpre_clear_source_config	configure timer OCPRE_CLR_INT source
timer_ocpre_clear_input_config	configure timer OCPRE_CLR input

Function name	Function description
timer_pulse_on_compare_config	configure the TIMER pulse on compare function
timer_dma_enable	enable the TIMER DMA
timer_dma_disable	disable the TIMER DMA
timer_channel_dma_request_source_select	channel DMA request source selection
timer_dma_transfer_config	configure the TIMER DMA transfer
timer_event_software_generate	software generate events
timer_break_struct_para_init	initialize the parameters of TIMER break parameter struct with the default values
timer_break_config	configure TIMER break function
timer_break_enable	enable TIMER break function
timer_break_disable	disable TIMER break function
timer_automatic_output_enable	enable TIMER output automatic function
timer_automatic_output_disable	disable TIMER output automatic function
timer_primary_output_config	configure TIMER primary output function
timer_channel_control_shadow_config	configure channel commutation control shadow register
timer_channel_control_shadow_update_config	configure TIMER channel control shadow register update control
timer_channel_output_struct_para_init	initialize the parameters of TIMER channel output parameter struct with the default values
timer_channel_output_config	configure TIMER channel output function
timer_channel_output_mode_config	configure TIMER channel output compare mode
timer_channel_output_pulse_value_config	configure TIMER channel output pulse value
timer_channel_output_pulse_fract_value_config	configure TIMER channel fractional part of 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_compare_fast_config	configure TIMER channel output compare fast 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

Function name	Function description
timer_channel_input_capture_prescaler_config	configure TIMER channel input capture prescaler value
timer_channel_capture_value_register_read	read TIMER channel capture compare register value
timer_input_pwm_capture_config	configure TIMER input pwm capture function
timer_hall_mode_config	configure TIMER hall sensor mode
timer_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_output0_trigger_source_select	select TIMER master mode output 0 trigger source
timer_master_output1_trigger_source_select	select TIMER master mode output 1 trigger source
timer_slave_mode_select	select TIMER slave mode
timer_pause_reset_polarity_config	configure TIMER pause+reset mode reset polarity
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_decoder_mode_config	configure TIMER decoder mode
timer_internal_clock_config	configure TIMER internal clock mode
timer_internal_trigger_as_external_clock_config	configure TIMER the internal trigger as external clock input
timer_external_trigger_as_external_clock_config	configure TIMER the external trigger as external clock input
timer_external_clock_mode0_config	configure TIMER the external clock mode 0
timer_external_clock_mode1_config	configure TIMER the external clock mode 1
timer_external_clock_mode1_disable	disable TIMER the external clock mode 1
timer_write_chxval_register_config	configure TIMER write CHxVAL register selection
timer_output_value_selection_config	configure TIMER output value selection
timer_commutation_control_shadow_register_config	configure commutation control shadow register update 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	configure the TIMER composite PWM mode output pulse

Function name	Function description
de_output_pulse_value_config	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_channel_additional_output_update_select	select TIMER channel additional output register update source
timer_channel_additional_compare_value_read	read TIMER channel additional compare value
timer_break_external_source_config	configure TIMER break external source
timer_break_external_polarity_config	configure TIMER break polarity
timer_break_lock_config	configure TIMER break lock function
timer_break_lock_release_config	configure the TIMER break lock release function
tiemr_dead_time_falling_edge_config	configure the TIMER falling edge dead time
timer_dead_time_different_config	configure the TIMER dead time different function
timer_dead_time_modify_config	configure the TIMER dead time modified on-the-fly function
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_parameter_init	initialize TIMER channel free complementary parameter struct with a default value
timer_channel_free_complementary_config	configure channel free complementary protection
timer_watchdog_value_config	configure quadrature decoder signal disconnection detection watchdog value
timer_watchdog_value_read	read quadrature decoder signal disconnection detection watchdog value
timer_decoder_disconnection_detection_config	configure quadrature decoder signal disconnection detection function
timer_decoder_jump_detection_config	configure decoder signal jump detection function
timer_decoder_modify_config	configure decoder mode modified on-the-fly function
timer_decoder_mode_update_source_config	configure decoder mode update source
timer_index_reset_counter_config	configure index signal reset counter function
timer_index_reset_direction_config	configure decoder index signal reset counter direction
timer_decoder_index_position_config	configure index position
timer_first_index_reset_counter_config	configure only the first index signal reset counter function

Function name	Function description
timer_upif_backup_config	configure the UPIF bit backup function
timer_upifbu_bit_get	get the UPIFBU bit in the TIMEx_CNT register
timer_counter_initial_register_config	configure counter initial value register
timer_counter_initial_config	configure counter initial value and direction
timer_synchronization_event_generate	generate soft synchronization event
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 3-1130. 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	the integer part of auto reload value, 0~0xFFFF (adjustment mode disable, TIMEx(x=0,2,3,5~7,14~16,19)) 0~0xFFFE(adjustment mode enable, TIMEx(x=0,2,3,5~7,14~16,19)) 0~0xFFFFFFFF(adjustment mode disable, TIMEx(x=1,4)) 0~0xFFFFFE(adjustment mode enable, TIMEx(x=1,4))
clockdivision	clock division value (TIMER_CKDIV_DIV1, TIMER_CKDIV_DIV2, TIMER_CKDIV_DIV4)
repetitioncounter	the counter repetition value (0~0xFF, use TIMER_CREP0 register; 0xFF~0xFFFFFFFF, use TIMER_CREP1 register)

### Structure timer\_break\_parameter\_struct

**Table 3-1131. 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)
outputautostate	output automatic enable (TIMER_OUTAUTO_ENABLE,

Member name	Function description
	TIMER_OUTAUTO_DISABLE)
protectmode	complementary register protect control(TIMER_CCHP0_PROT_OFF, TIMER_CCHP0_PROT_0, TIMER_CCHP0_PROT_1, TIMER_CCHP0_PROT_2)
break0state	BREAK0 input enable (TIMER_BREAK0_ENABLE, TIMER_BREAK0_DISABLE)
break0filter	BREAK0 input filter(0~15)
break0polarity	BREAK0 input polarity(TIMER_BREAK0_POLARITY_LOW, TIMER_BREAK0_POLARITY_HIGH)
break0lock	BREAK0 input lock(TIMER_BREAK0_LK_ENABLE, TIMER_BREAK0_LK_DISABLE)
break0release	BREAK0 input release(TIMER_BREAK0_RELEASE, TIMER_BREAK0_UNRELEASE)
break1state	BREAK1 input enable (TIMER_BREAK1_ENABLE, TIMER_BREAK1_DISABLE)
break1filter	BREAK1 input filter(0~15)
break1polarity	BREAK1 input polarity(TIMER_BREAK1_POLARITY_LOW, TIMER_BREAK1_POLARITY_HIGH)
break1lock	BREAK1 input lock(TIMER_BREAK1_LK_ENABLE, TIMER_BREAK1_LK_DISABLE)
break1release	BREAK1 input release(TIMER_BREAK1_RELEASE, TIMER_BREAK1_UNRELEASE)

### Structure timer\_oc\_parameter\_struct

**Table 3-1132. 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 3-1133. Structure timer\_omc\_parameter\_struct**

Member name	Function description
outputmode	multi mode channel output mode selection(TIMER_MCH_MODE_INDEPENDENTLY, 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 3-1134. Structure timer\_ic\_parameter\_struct**

Member name	Function description
icpolarity	channel input polarity(TIMER_IC_POLARITY_RISING, TIMER_IC_POLARITY_FALLING, TIMER_IC_POLARITY_BOTH_EDGE)
icselection	channel input mode selection(TIMER_IC_SELECTION_DIRECTTI, TIMER_IC_SELECTION_INDIRECTTI, TIMER_IC_SELECTION_ITS, 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\_free\_complementary\_parameter\_struct

**Table 3-1135. 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)
fallingdeadtime	falling edge dead time(0~255)

## timer\_deinit

The description of timer\_deinit is shown as below:

**Table 3-1136. Function timer\_deinit**

Function name	timer_deinit
---------------	--------------

<b>Function prototype</b>	void timer_deinit(uint32_t timer_periph);
<b>Function descriptions</b>	deinit a TIMER
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset TIMER0 */
timer_deinit(TIMER0);
```

### timer\_struct\_para\_init

The description of timer\_struct\_para\_init is shown as below:

**Table 3-1137. Function timer\_struct\_para\_init**

<b>Function name</b>	timer_struct_para_init
<b>Function prototype</b>	void timer_struct_para_init(timer_parameter_struct* initpara);
<b>Function descriptions</b>	initialize the parameters of TIMER init parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>initpara</b>	TIMER init parameter struct, the structure members can refer to <a href="#">Structure timer_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1138. Function timer\_init**

<b>Function name</b>	timer_init
<b>Function prototype</b>	void timer_init(uint32_t timer_periph, timer_parameter_struct* initpara);
<b>Function descriptions</b>	initialize TIMER counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>initpara</b>	TIMER init parameter struct, the structure members can refer to <a href="#">Structure timer_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1139. Function timer\_enable**

<b>Function name</b>	timer_enable
<b>Function prototype</b>	void timer_enable(uint32_t timer_periph);

<b>Function descriptions</b>	enable a TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TIMER0 */
timer_enable(TIMER0);
```

### timer\_disable

The description of timer\_disable is shown as below:

**Table 3-1140. Function timer\_disable**

<b>Function name</b>	timer_disable
<b>Function prototype</b>	void timer_disable(uint32_t timer_periph);
<b>Function descriptions</b>	disable a TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TIMER0 */
timer_disable(TIMER0);
```

### timer\_auto\_reload\_shadow\_enable

The description of timer\_auto\_reload\_shadow\_enable is shown as below:

Table 3-1141. Function timer\_auto\_reload\_shadow\_enable

<b>Function name</b>	timer_auto_reload_shadow_enable
<b>Function prototype</b>	void timer_auto_reload_shadow_enable(uint32_t timer_periph);
<b>Function descriptions</b>	enable the auto reload shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1142. Function timer\_auto\_reload\_shadow\_disable

<b>Function name</b>	timer_auto_reload_shadow_disable
<b>Function prototype</b>	void timer_auto_reload_shadow_disable (uint32_t timer_periph);
<b>Function descriptions</b>	disable the auto reload shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1143. Function timer\_update\_event\_enable**

<b>Function name</b>	timer_update_event_enable
<b>Function prototype</b>	void timer_update_event_enable(uint32_t timer_periph);
<b>Function descriptions</b>	enable the update event
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~7, 14~16, 19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1144. Function timer\_update\_event\_disable**

<b>Function name</b>	timer_update_event_disable
<b>Function prototype</b>	void timer_update_event_disable(uint32_t timer_periph);
<b>Function descriptions</b>	disable the update event
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~7, 14~16, 19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1145. Function timer\_counter\_alignment**

<b>Function name</b>	timer_counter_alignment
<b>Function prototype</b>	void timer_counter_alignment(uint32_t timer_periph, uint16_t aligned);
<b>Function descriptions</b>	set TIMER counter alignment mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>aligned</b>	alignment mode
<i>TIMER_COUNTER_EDGE</i>	edge-aligned mode
<i>TIMER_COUNTER_CENTR_DOWN</i>	center-aligned and counting down assert mode
<i>TIMER_COUNTER_CENTR_UP</i>	center-aligned and counting up assert mode
<i>TIMER_COUNTER_CENTR_BOTH</i>	center-aligned and counting up/down assert mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set TIMER0 counter center-aligned and counting up assert mode */
```

```
timer_counter_alignment(TIMER0, TIMER_COUNTER_CENTR_UP);
```

### timer\_counter\_up\_direction

The description of timer\_counter\_up\_direction is shown as below:

**Table 3-1146. Function timer\_counter\_up\_direction**

<b>Function name</b>	timer_counter_up_direction
<b>Function prototype</b>	void timer_counter_up_direction(uint32_t timer_periph);
<b>Function descriptions</b>	set TIMER counter up direction
<b>Precondition</b>	set TIMERx counter no center-aligned mode (edge-aligned mode)
<b>The called functions</b>	-

Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</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 3-1147. Function timer\_counter\_down\_direction**

<b>Function name</b>	timer_counter_down_direction
<b>Function prototype</b>	void timer_counter_down_direction(uint32_t timer_periph);
<b>Function descriptions</b>	set TIMER counter down direction
<b>Precondition</b>	set TIMERx counter no center-aligned mode (edge-aligned mode)
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set TIMER0 counter down direction */
```

```
timer_counter_down_direction(TIMER0);
```

### timer\_adjustment\_mode\_config

The description of timer\_adjustment\_mode\_config is shown as below:

**Table 3-1148. Function timer\_adjustment\_mode\_config**

<b>Function name</b>	timer_adjustment_mode_config
<b>Function prototype</b>	void timer_adjustment_mode_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure the TIMER adjustment mode function

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TIMER adjustment mode function */
```

```
timer_adjustment_mode_config(TIMER0, ENABLE);
```

### timer\_prescaler\_config

The description of timer\_prescaler\_config is shown as below:

**Table 3-1149. Function timer\_prescaler\_config**

<b>Function name</b>	timer_prescaler_config
<b>Function prototype</b>	void timer_prescaler_config(uint32_t timer_periph, uint16_t prescaler, uint32_t pscreload);
<b>Function descriptions</b>	configure TIMER prescaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>prescaler</b>	prescaler value (0~0xFFFF)
<b>Input parameter{in}</b>	
<b>pscreload</b>	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
<b>Output parameter{out}</b>	

-	-
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 3-1150. Function timer\_repetition\_value\_config**

<b>Function name</b>	timer_repetition_value_config
<b>Function prototype</b>	void timer_repetition_value_config(uint32_t timer_periph, uint16_t ccsel, uint32_t repetition);
<b>Function descriptions</b>	configure TIMER repetition register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>ccsel</b>	repetition register selection
<i>TIMER_CREP0_ENABLE</i>	the update event rate is depended to TIMERx_CREP0 register
<i>TIMER_CREP1_ENABLE</i>	the update event rate is depended to TIMERx_CREP1 register
<b>Input parameter{in}</b>	
<b>repetition</b>	the counter repetition value (0~0xFF, use TIMER_CREP0 register; 0~0xFFFFFFFF, use TIMER_CREP1 register)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 repetition register 0 value */
```

```
timer_repetition_value_config(TIMER0, TIMER_CREP0_ENABLE, 98);
```



## timer\_runtime\_repetition\_value\_read

The description of timer\_runtime\_repetition\_value\_read is shown as below:

**Table 3-1151. Function timer\_runtime\_repetition\_value\_read**

<b>Function name</b>	timer_runtime_repetition_value_read
<b>Function prototype</b>	uint32_t timer_runtime_repetition_value_read(uint32_t timer_periph);
<b>Function descriptions</b>	read TIMER runtime repetition register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,19)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	counter repetition value in TIMER_CREP1 register (0~0xFFFFFFFF)

Example:

```
/* read TIMER0 runtime repetition register value */
uint32_t i = 0;
i = timer_runtime_repetition_value_read(TIMER0);
```

## timer\_autoreload\_value\_config

The description of timer\_autoreload\_value\_config is shown as below:

**Table 3-1152. Function timer\_autoreload\_value\_config**

<b>Function name</b>	timer_autoreload_value_config
<b>Function prototype</b>	void timer_autoreload_value_config(uint32_t timer_periph, uint64_t autoreload);
<b>Function descriptions</b>	configure TIMER autoreload register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>autoreload</b>	the integer part of auto-reload value, 0~0xFFFF(adjustment mode disable, TIMERx(x=0,2,3,5~7,14~16,19)) 0~0xFFFE(adjustment mode enable, TIMERx(x=0,2,3,5~7,14~16,19))

	0~0xFFFFFFFF(adjustment mode disable, TIMEx(x=1,4)) 0~0xFFFFFEE(adjustment mode enable, TIMEx(x=1,4))
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER autoreload register value */
```

```
timer_autoreload_value_config(TIMERO, 3000);
```

### timer\_auto\_reload\_fract\_value\_config

The description of timer\_auto\_reload\_fract\_value\_config is shown as below:

**Table 3-1153. Function timer\_auto\_reload\_fract\_value\_config**

<b>Function name</b>	timer_auto_reload_fract_value_config
<b>Function prototype</b>	void timer_auto_reload_fract_value_config(uint32_t timer_periph, uint32_t fract);
<b>Function descriptions</b>	configure TIMER fractional part of auto reload register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMEx(x=0~7, 14~16, 19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>fract</b>	fractional part of auto reload value, 0~F
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER fractional part of auto reload register value */
```

```
timer_auto_reload_fract_value_config (TIMERO, 15);
```

### timer\_autoreload\_value\_read

The description of timer\_autoreload\_value\_read is shown as below:

**Table 3-1154. Function timer\_autoreload\_value\_read**

<b>Function name</b>	timer_autoreload_value_read
<b>Function prototype</b>	uint32_t timer_autoreload_value_read(uint32_t timer_periph);

<b>Function descriptions</b>	read TIMER autoreload register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 19)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the integer part of auto-reload value, 0~0xFFFF (adjustment mode disable, <i>TIMERx(x=0,2,3,5~7,14~16,19)</i> ) 0~0xFFFFE (adjustment mode enable, <i>TIMERx(x=0,2,3,5~7,14~16,19)</i> ) 0~0xFFFFFFFF (adjustment mode disable, <i>TIMERx(x=1,4)</i> ) 0~0xFFFFF (adjustment mode enable, <i>TIMERx(x=1,4)</i> )

Example:

```
/* get TIMER autoreload register value */
uint32_t i = 0;
i =(uint32_t) timer_autoreload_value_read (TIMER0);
```

### timer\_auto\_reload\_fract\_value\_read

The description of timer\_auto\_reload\_fract\_value\_read is shown as below:

**Table 3-1155. Function timer\_auto\_reload\_fract\_value\_read**

<b>Function name</b>	timer_auto_reload_fract_value_read
<b>Function prototype</b>	uint32_t timer_auto_reload_fract_value_read(uint32_t timer_periph);
<b>Function descriptions</b>	read TIMER fractional part of auto reload register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 19)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	fractional part of auto reload value, 0~F

Example:

```
/* get TIMER fractional part of auto reload register value */
```

```
uint32_t i = 0;
```

```
i=(uint32_t) timer_auto_reload_fract_value_read (TIMER0);
```

## timer\_counter\_value\_config

The description of timer\_counter\_value\_config is shown as below:

**Table 3-1156. Function timer\_counter\_value\_config**

<b>Function name</b>	timer_counter_value_config
<b>Function prototype</b>	void timer_counter_value_config(uint32_t timer_periph, uint32_t counter);
<b>Function descriptions</b>	configure TIMER counter register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>counter</b>	the counter value 0~0xFFFF, TIMERx(x=0,2,3,5~7,14~16,19) 0~0xFFFFFFFF, TIMERx(x=1,4)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1157. Function timer\_counter\_read**

<b>Function name</b>	timer_counter_read
<b>Function prototype</b>	uint32_t timer_counter_read(uint32_t timer_periph);
<b>Function descriptions</b>	read TIMER counter value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 19)</i>	TIMER peripheral selection

Output parameter{out}	
-	-
Return value	
uint32_t	counter value 0~0xFFFF, TIMERx(x=0,2,3,5~7,14~16,19) 0~0xFFFFFFFF, TIMERx(x=1,4)

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 3-1158. Function timer\_prescaler\_read**

Function name	timer_prescaler_read
Function prototype	uint16_t timer_prescaler_read(uint32_t timer_periph);
Function descriptions	read TIMER prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 19)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
uint16_t	prescaler register value (0~0xFFFF)

Example:

```
/* read TIMER0 prescaler value */

uint16_t i = 0;

i = timer_prescaler_read(TIMER0);
```

### timer\_single\_pulse\_mode\_config

The description of timer\_single\_pulse\_mode\_config is shown as below:

**Table 3-1159. 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);
<b>Function descriptions</b>	configure TIMER single pulse mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>spmode</b>	pulse mode
<i>TIMER_SP_MODE_SINGLE</i>	single pulse mode
<i>TIMER_SP_MODE_REPETITIVE</i>	repetitive pulse mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 single pulse mode */
```

```
timer_single_pulse_mode_config(TIMER0, TIMER_SP_MODE_SINGLE);
```

### timer\_delayable\_single\_pulse\_mode\_config

The description of timer\_delayable\_single\_pulse\_mode\_config is shown as below:

**Table 3-1160. Function timer\_delayable\_single\_pulse\_mode\_config**

<b>Function name</b>	timer_delayable_single_pulse_mode_config
<b>Function prototype</b>	void timer_delayable_single_pulse_mode_config(uint32_t timer_periph, uint16_t channel, uint32_t dspmode, uint16_t cnt_dir);
<b>Function descriptions</b>	configure timer delayable single pulse mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx(x=0~4,7,14,19))
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx(x=0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx(x=0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx(x=0~4,7,19))

<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx(x=0~4,7,14,19))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1(TIMERx(x=0,7,19))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2(TIMERx(x=0,7,19))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3(TIMERx(x=0,7,19))
<b>Input parameter{in}</b>	
<b>dspmode</b>	delayable SPM mode
<i>TIMER_OC_MODE_DS</i> <i>PM0</i>	delayable SPM mode 0
<i>TIMER_OC_MODE_DS</i> <i>PM1</i>	delayable SPM mode1
<b>Input parameter{in}</b>	
<b>cnt_dir</b>	counter direction selection
<i>TIMER_COUNTER_UP</i>	count up
<i>TIMER_COUNTER_DO</i> <i>WN</i>	count down
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0_CH0 delayable single pulse mode */
```

```
timer_delayable_single_pulse_mode_config(TIMER0,                                TIMER_CH_0,
TIMER_OC_MODE_DSPM0, TIMER_COUNTER_UP);
```

### timer\_update\_source\_config

The description of timer\_update\_source\_config is shown as below:

**Table 3-1161. Function timer\_update\_source\_config**

<b>Function name</b>	timer_update_source_config
<b>Function prototype</b>	void timer_update_source_config(uint32_t timer_periph, uint32_t update);
<b>Function descriptions</b>	configure TIMER update source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>update</b>	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

	<ul style="list-style-type: none"> <li>– The counter generates an overflow or underflow event</li> <li>– The slave mode controller generates an update event</li> </ul>
<i>TIMER_UPDATE_SRC_REGULAR</i>	only counter overflow/underflow generates an update interrupt or DMA request.
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER update only by counter overflow/underflow */
```

```
timer_update_source_config(TIMER0, TIMER_UPDATE_SRC_REGULAR);
```

### timer\_ocpre\_clear\_source\_config

The description of timer\_ocpre\_clear\_source\_config is shown as below:

**Table 3-1162. Function timer\_ocpre\_clear\_source\_config**

<b>Function name</b>	timer_ocpre_clear_source_config
<b>Function prototype</b>	void timer_ocpre_clear_source_config(uint32_t timer_periph, uint32_t clear_source);
<b>Function descriptions</b>	configure timer OCPRE_CLR_INT source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>clear_source</b>	OCPRE_CLR_INT source
<i>TIMER_OCPRE_CLR_INT_OCPRE_CLR</i>	OCPRE_CLR_INT is connected to the OCPRE_CLR input
<i>TIMER_OCPRE_CLR_INT_ETIF</i>	OCPRE_CLR_INT is connected to ETIF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure timer OCPRE_CLR_INT source */
```

```
timer_ocpre_clear_source_config (TIMER0, TIMER_OCPRE_CLR_INT_ETIF);
```



## timer\_ocpre\_clear\_input\_config

The description of timer\_ocpre\_clear\_input\_config is shown as below:

**Table 3-1163. Function timer\_ocpre\_clear\_input\_config**

<b>Function name</b>	timer_ocpre_clear_input_config
<b>Function prototype</b>	void timer_ocpre_clear_input_config(uint32_t timer_periph, uint32_t clear_input);
<b>Function descriptions</b>	configure timer OCPRE_CLR input
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>clear_source</b>	OCPRE_CLR source
<i>TIMER_OCPRE_CLR_INPUT_x</i>	OCPRE_CLR_INT is connected to OCPRE_CLR x(x=0~7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure timer OCPRE_CLR input */
timer_ocpre_clear_input_config (TIMER0, TIMER_OCPRE_CLR_INPUT_7);
```

## timer\_pulse\_on\_compare\_config

The description of timer\_pulse\_on\_compare\_config is shown as below:

**Table 3-1164. Function timer\_pulse\_on\_compare\_config**

<b>Function name</b>	timer_pulse_on_compare_config
<b>Function prototype</b>	void timer_ocpre_clear_input_config(uint32_t timer_periph, uint32_t clear_input);
<b>Function descriptions</b>	configure the TIMER pulse on compare function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>pulse_width</b>	output pulse width, 0~255
<b>Input parameter{in}</b>	

<b>pulse_prescaler</b>	output pulse prescaler, 0~7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER pulse on compare function */
```

```
timer_pulse_on_compare_config (TIMER0, 100, 7);
```

### timer\_dma\_enable

The description of timer\_dma\_enable is shown as below:

**Table 3-1165. Function timer\_dma\_enable**

<b>Function name</b>	timer_dma_enable
<b>Function prototype</b>	void timer_dma_enable(uint32_t timer_periph, uint32_t dma);
<b>Function descriptions</b>	enable the TIMER DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>dma</b>	timer DMA source enable
<i>TIMER_DMA_UPD</i>	update DMA request, TIMERx(x=0~7,14~16,19)
<i>TIMER_DMA_CH0D</i>	channel 0 DMA request, TIMERx (x=0~4,7,14~16,19)
<i>TIMER_DMA_CH1D</i>	channel 1 DMA request, TIMERx (x=0~4,7,14,19)
<i>TIMER_DMA_CH2D</i>	channel 2 DMA request, TIMERx (x=0~4,7,19)
<i>TIMER_DMA_CH3D</i>	channel 3 DMA request, TIMERx (x=0~4,7,19)
<i>TIMER_DMA_CMTD</i>	commutation DMA request, TIMERx (x=0,7,14~16,19)
<i>TIMER_DMA_TRGD</i>	trigger DMA request, TIMERx (x=0~4,7,14,19)
<i>TIMER_DMA_MCH0D</i>	multi mode channel 0 DMA request, TIMERx (x=0,7,14~16,19)
<i>TIMER_DMA_MCH1D</i>	multi mode channel 1 DMA request, TIMERx (x=0,7,19)
<i>TIMER_DMA_MCH2D</i>	multi mode channel 2 DMA request, TIMERx (x=0,7,19)
<i>TIMER_DMA_MCH3D</i>	multi mode channel 3 DMA request, TIMERx (x=0,7,19)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1166. Function timer\_dma\_disable**

<b>Function name</b>	timer_dma_disable
<b>Function prototype</b>	void timer_dma_disable (uint32_t timer_periph, uint32_t dma);
<b>Function descriptions</b>	disable the TIMER DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>dma</b>	timer DMA source enable
<i>TIMER_DMA_UPD</i>	update DMA request, <i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)
<i>TIMER_DMA_CH0D</i>	channel 0 DMA request, <i>TIMERx</i> ( <i>x</i> =0~4, 7, 14~16, 19)
<i>TIMER_DMA_CH1D</i>	channel 1 DMA request, <i>TIMERx</i> ( <i>x</i> =0~4, 7, 14, 19)
<i>TIMER_DMA_CH2D</i>	channel 2 DMA request, <i>TIMERx</i> ( <i>x</i> =0~4, 7, 19)
<i>TIMER_DMA_CH3D</i>	channel 3 DMA request, <i>TIMERx</i> ( <i>x</i> =0~4, 7, 19)
<i>TIMER_DMA_CMTD</i>	commutation DMA request, <i>TIMERx</i> ( <i>x</i> =0, 7, 14~16, 19)
<i>TIMER_DMA_TRGD</i>	trigger DMA request, <i>TIMERx</i> ( <i>x</i> =0~4, 7, 14, 19)
<i>TIMER_DMA_MCH0D</i>	multi mode channel 0 DMA request, <i>TIMERx</i> ( <i>x</i> =0, 7, 14~16, 19)
<i>TIMER_DMA_MCH1D</i>	multi mode channel 1 DMA request, <i>TIMERx</i> ( <i>x</i> =0, 7, 19)
<i>TIMER_DMA_MCH2D</i>	multi mode channel 2 DMA request, <i>TIMERx</i> ( <i>x</i> =0, 7, 19)
<i>TIMER_DMA_MCH3D</i>	multi mode channel 3 DMA request, <i>TIMERx</i> ( <i>x</i> =0, 7, 19)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1167. Function timer\_channel\_dma\_request\_source\_select

<b>Function name</b>	timer_channel_dma_request_source_select
<b>Function prototype</b>	void timer_channel_dma_request_source_select(uint32_t timer_periph, uint32_t dma_request);
<b>Function descriptions</b>	channel DMA request source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4, 7, 14~16, 19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>dma_request</b>	channel DMA request source selection
<i>TIMER_DMAREQUEST_CHANNELEVENT</i>	DMA request of channel n is sent when channel n event occurs
<i>TIMER_DMAREQUEST_UPDATEEVENT</i>	DMA request of channel n is sent when update event occurs
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1168. Function timer\_dma\_transfer\_config

<b>Function name</b>	timer_dma_transfer_config
<b>Function prototype</b>	void timer_dma_transfer_config(uint32_t timer_periph, uint32_t dma_baseaddr, uint32_t dma_lenth);
<b>Function descriptions</b>	configure the TIMER DMA transfer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4, 7, 14~16, 19)	please refer to the following parameters
<b>Input parameter{in}</b>	

<b>dma_baseaddr</b>	DMA transfer access start address
<i>TIMER_DMACFG_DMA</i> <i>TA_CTL0</i>	DMA transfer address is <i>TIMER_CTL0</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CTL1</i>	DMA transfer address is <i>TIMER_CTL1</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_SMCFG</i>	DMA transfer address is <i>TIMER_SMCFG</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_DMAINTEN</i>	DMA transfer address is <i>TIMER_DMAINTEN</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_INTF</i>	DMA transfer address is <i>TIMER_INTF</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_SWEVG</i>	DMA transfer address is <i>TIMER_SWEVG</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CHCTL0</i>	DMA transfer address is <i>TIMER_CHCTL0</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CHCTL1</i>	DMA transfer address is <i>TIMER_CHCTL1</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CHCTL2</i>	DMA transfer address is <i>TIMER_CHCTL2</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CNT</i>	DMA transfer address is <i>TIMER_CNT</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_PSC</i>	DMA transfer address is <i>TIMER_PSC</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CAR</i>	DMA transfer address is <i>TIMER_CAR</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CREP0</i>	DMA transfer address is <i>TIMER_CREP0</i> , <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH0CV</i>	DMA transfer address is <i>TIMER_CH0CV</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH1CV</i>	DMA transfer address is <i>TIMER_CH1CV</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH2CV</i>	DMA transfer address is <i>TIMER_CH2CV</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH3CV</i>	DMA transfer address is <i>TIMER_CH3CV</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_CCHP0</i>	DMA transfer address is <i>TIMER_CCHP0</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL0</i>	DMA transfer address is <i>TIMER_MCHCTL0</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL1</i>	DMA transfer address is <i>TIMER_MCHCTL1</i> , <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL2</i>	DMA transfer address is <i>TIMER_MCHCTL2</i> , <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)

<i>TIMER_DMACFG_DMA TA_MCH0CV</i>	DMA transfer address is TIMER_MCH0CV, TIMERx(x=0,7,14~16,19)
<i>TIMER_DMACFG_DMA TA_MCH1CV</i>	DMA transfer address is TIMER_MCH1CV, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_MCH2CV</i>	DMA transfer address is TIMER_TIMER_MCH2CV, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_MCH3CV</i>	DMA transfer address is TIMER_TIMER_MCH3CV, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_CH0COMV_ADD</i>	DMA transfer address is TIMER_CH0COMV_ADD, TIMERx(x=0~4,7,14,19)
<i>TIMER_DMACFG_DMA TA_CH1COMV_ADD</i>	DMA transfer address is TIMER_CH1COMV_ADD, TIMERx(x=0~4,7,14,19)
<i>TIMER_DMACFG_DMA TA_CH2COMV_ADD</i>	DMA transfer address is TIMER_CH2COMV_ADD, TIMERx(x=0~4,7,19)
<i>TIMER_DMACFG_DMA TA_CH3COMV_ADD</i>	DMA transfer address is TIMER_CH3COMV_ADD, TIMERx(x=0~4,7,19)
<i>TIMER_DMACFG_DMA TA_CTL2</i>	DMA transfer address is TIMER_CTL2, TIMERx(x=0~4,7,14~16,19)
<i>TIMER_DMACFG_DMA TA_FCCHP0</i>	DMA transfer address is TIMER_FCCHP0, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_FCCHP1</i>	DMA transfer address is TIMER_FCCHP1, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_FCCHP2</i>	DMA transfer address is TIMER_FCCHP2, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_FCCHP3</i>	DMA transfer address is TIMER_FCCHP3, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_AFCTL0</i>	DMA transfer address is TIMER_AFCTL0, TIMERx(x=0,7,14~16,19)
<i>TIMER_DMACFG_DMA TA_AFCTL1</i>	DMA transfer address is TIMER_AFCTL1, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_WDGCNT</i>	DMA transfer address is TIMER_WDGCNT, TIMERx(x=0~4,7,19)
<i>TIMER_DMACFG_DMA TA_CREP1</i>	DMA transfer address is TIMER_CREP1, TIMERx(x=0,7,14~16,19)
<i>TIMER_DMACFG_DMA TA_CCHP1</i>	DMA transfer address is TIMER_CCHP1, TIMERx(x=0,7,14~16,19)
<i>TIMER_DMACFG_DMA TA_DECCTL</i>	DMA transfer address is TIMER_DECCTL, TIMERx(0~4,7,19)
<i>TIMER_DMACFG_DMA TA_CINITCTL</i>	DMA transfer address is TIMER_CINITCTL, TIMERx(x=0,7,19)
<i>TIMER_DMACFG_DMA TA_CINITV</i>	DMA transfer address is TIMER_CINITV, TIMERx(x=0,7,19)
<b>Input parameter{in}</b>	

<b>dma_lenth</b>	DMA transfer count
<i>TIMER_DMACFG_DMA TC_xTRANSFER</i>	(x=1~42), DMA transfer x time
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1169. Function timer\_event\_software\_generate**

<b>Function name</b>	timer_event_software_generate
<b>Function prototype</b>	void timer_event_software_generate(uint32_t timer_periph, uint32_t event);
<b>Function descriptions</b>	software generate events
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>event</b>	the timer software event generation sources
<i>TIMER_EVENT_SRC_UPDATE</i>	update event, TIMERx(x=0~7, 14~16, 19)
<i>TIMER_EVENT_SRC_C0H0G</i>	channel 0 capture or compare event generation, TIMERx(x=0~4, 7, 14~16, 19)
<i>TIMER_EVENT_SRC_C1H1G</i>	channel 1 capture or compare event generation, TIMERx(x=0~4, 7, 14, 19)
<i>TIMER_EVENT_SRC_C2H2G</i>	channel 2 capture or compare event generation, TIMERx(x=0~4, 7, 19)
<i>TIMER_EVENT_SRC_C3H3G</i>	channel 3 capture or compare event generation, TIMERx(x=0~4, 7, 19)
<i>TIMER_EVENT_SRC_CMTG</i>	channel commutation event generation, TIMERx(x=0, 7, 14~16, 19)
<i>TIMER_EVENT_SRC_TRIGGER</i>	trigger event generation, TIMERx(x=0~4, 7, 14, 19)

<i>RGG</i>	
<i>TIMER_EVENT_SRC_B RK0G</i>	BREAK0 event generation, $TIMERx(x=0,7,14\sim16,19)$
<i>TIMER_EVENT_SRC_B RK1G</i>	BREAK1 event generation, $TIMERx(x=0,7,19)$
<i>TIMER_EVENT_SRC_M CH0G</i>	multi mode channel 0 capture or compare event generation, $TIMERx(x=0,7,14\sim16,19)$
<i>TIMER_EVENT_SRC_M CH1G</i>	multi mode channel 1 capture or compare event generation, $TIMERx(x=0,7,19)$
<i>TIMER_EVENT_SRC_M CH2G</i>	multi mode channel 2 capture or compare event generation, $TIMERx(x=0,7,19)$
<i>TIMER_EVENT_SRC_M CH3G</i>	multi mode channel 3 capture or compare event generation, $TIMERx(x=0,7,19)$
<i>TIMER_EVENT_SRC_C H0COMADDG</i>	channel 0 additional compare event generation, $TIMERx(x=0\sim4,7,14,19)$
<i>TIMER_EVENT_SRC_C H1COMADDG</i>	channel 1 additional compare event generation, $TIMERx(x=0\sim4,7,14,19)$
<i>TIMER_EVENT_SRC_C H2COMADDG</i>	channel 2 additional compare event generation, $TIMERx(x=0\sim4,7,19)$
<i>TIMER_EVENT_SRC_C H3COMADDG</i>	channel 3 additional compare event generation, $TIMERx(x=0\sim4,7,19)$
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1170. Function timer\_break\_struct\_para\_init**

<b>Function name</b>	timer_break_struct_para_init
<b>Function prototype</b>	void timer_break_struct_para_init(timer_break_parameter_struct* breakpara);
<b>Function descriptions</b>	initialize the parameters of TIMER break parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	



<b>breakpara</b>	TIMER break parameter struct, the structure members can refer to <a href="#">Structure timer break parameter struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1171. Function timer\_break\_config**

<b>Function name</b>	timer_break_config
<b>Function prototype</b>	void timer_break_config(uint32_t timer_periph, timer_break_parameter_struct* breakpara);
<b>Function descriptions</b>	configure TIMER break function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>breakpara</b>	TIMER break parameter struct, the structure members can refer to <a href="#">Structure timer break parameter struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
timer_break_parameter_struct timer_breakpara;
```

```
timer_breakpara.runoffstate = TIMER_ROS_STATE_DISABLE;
```

```
timer_breakpara.ideloffstate = TIMER_IOS_STATE_DISABLE;
```

```
timer_breakpara.deadtime = 0U;
```

```
timer_breakpara.outputautostate = TIMER_OUTAUTO_ENABLE;
```

```

timer_breakpara.protectmode      = TIMER_CCHP0_PROT_OFF;

timer_breakpara.break0state      = TIMER_BREAK0_ENABLE;

timer_breakpara.break0filter     = 0U;

timer_breakpara.break0polarity   = TIMER_BREAK0_POLARITY_LOW;

timer_breakpara.break0bidirectional = TIMER_BREAK0_LK_ENABLE;

timer_breakpara.break0release    = TIMER_BREAK0_UNRELEASE;

timer_breakpara.break1state      = TIMER_BREAK1_DISABLE;

timer_breakpara.break1filter     = 0U;

timer_breakpara.break1polarity   = TIMER_BREAK1_POLARITY_LOW;

timer_breakpara.break1bidirectional = TIMER_BREAK1_LK_DISABLE;

timer_breakpara.break1release    = TIMER_BREAK1_UNRELEASE;

timer_break_config(TIMER0, &timer_breakpara);

```

### timer\_break\_enable

The description of timer\_break\_enable is shown as below:

**Table 3-1172. Function timer\_break\_enable**

Function name	timer_break_enable
Function prototype	void timer_break_enable(uint32_t timer_periph, uint16_t break_num);
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}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	TIMER peripheral selection
Input parameter{in}	
<b>break_num</b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	BREAK0 input signals, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_BREAK1</i>	BREAK1 input signals, <i>TIMERx</i> ( <i>x</i> =0,7,19)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 BREAK0 function*/
```

```
timer_break_enable(TIMER0, TIMER_BREAK0);
```

### timer\_break\_disable

The description of timer\_break\_disable is shown as below:

**Table 3-1173. Function timer\_break\_disable**

<b>Function name</b>	timer_break_disable
<b>Function prototype</b>	void timer_break_disable(uint32_t timer_periph, uint16_t break_num);
<b>Function descriptions</b>	disable TIMER break function
<b>Precondition</b>	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>break_num</b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	BREAK0 input signals, TIMERx(x=0,7,14~16,19)
<i>TIMER_BREAK1</i>	BREAK1 input signals, TIMERx(x=0,7,19)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TIMER0 BREAK0 function*/
```

```
timer_break_disable(TIMER0, TIMER_BREAK0);
```

### timer\_automatic\_output\_enable

The description of timer\_automatic\_output\_enable is shown as below:

**Table 3-1174. Function timer\_automatic\_output\_enable**

<b>Function name</b>	timer_automatic_output_enable
<b>Function prototype</b>	void timer_automatic_output_enable(uint32_t timer_periph);
<b>Function descriptions</b>	enable TIMER output automatic function
<b>Precondition</b>	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TIMER0 output automatic function */
```

```
timer_automatic_output_enable(TIMER0);
```

### timer\_automatic\_output\_disable

The description of timer\_automatic\_output\_disable is shown as below:

**Table 3-1175. Function timer\_automatic\_output\_disable**

<b>Function name</b>	timer_automatic_output_disable
<b>Function prototype</b>	void timer_automatic_output_disable (uint32_t timer_periph);
<b>Function descriptions</b>	disable TIMER output automatic function
<b>Precondition</b>	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1176. Function timer\_primary\_output\_config**

<b>Function name</b>	timer_primary_output_config
<b>Function prototype</b>	void timer_primary_output_config(uint32_t timer_periph, ControlStatus newvalue);

<b>Function descriptions</b>	configure TIMER primary output function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TIMER0 primary output function */
```

```
timer_primary_output_config(TIMER0, ENABLE);
```

### timer\_channel\_control\_shadow\_config

The description of timer\_channel\_control\_shadow\_config is shown as below:

**Table 3-1177. Function timer\_channel\_control\_shadow\_config**

<b>Function name</b>	timer_channel_control_shadow_config
<b>Function prototype</b>	void timer_channel_control_shadow_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure channel commutation control shadow register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1178. 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}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,19)</i>	TIMER peripheral selection
Input parameter{in}	
<b>ccuctl</b>	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
<i>TIMER_UPDATECTL_CCUOVER</i>	the shadow registers update by when the overflow event occurs
<i>TIMER_UPDATECTL_CCUUNDER</i>	the shadow registers update by when the underflow event occurs
<i>TIMER_UPDATECTL_CCUOVERUNDER</i>	the shadow registers update by when the overflow or underflow event occurs
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel control shadow register update when CMTG bit is set */
timer_channel_control_shadow_update_config(TIMER0, TIMER_UPDATECTL_CCU);
```

### timer\_channel\_output\_struct\_para\_init

The description of timer\_channel\_output\_struct\_para\_init is shown as below:

Table 3-1179. 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 <a href="#">Structure timer_oc_parameter_struct</a> .
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 3-1180. 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=0~4,7,14~16,19)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
TIMER_CH_2	TIMER channel 2(TIMERx(x=0~4,7,19))
TIMER_CH_3	TIMER channel 3(TIMERx(x=0~4,7,19))
Input parameter{in}	

<b>ocpara</b>	TIMER channel output parameter struct, the structure members can refer to <a href="#">Structure timer oc parameter struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1181. Function timer\_channel\_output\_mode\_config**

<b>Function name</b>	timer_channel_output_mode_config
<b>Function prototype</b>	void timer_channel_output_mode_config(uint32_t timer_periph, uint16_t channel, uint32_t ocmode);
<b>Function descriptions</b>	configure TIMER channel output compare mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4,7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4,7,19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4,7,19))
TIMER_MCH_0	TIMER multi mode channel 0 (TIMERx(x=0,7,14~16,19))



<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7,19))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7,19))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7,19))
<b>Input parameter{in}</b>	
<b>ocmode</b>	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
<i>TIMER_OC_MODE_DSPM0</i>	delayable SPM mode 0
<i>TIMER_OC_MODE_DSPM1</i>	delayable SPM mode 1
<i>TIMER_OC_MODE_PULSE</i>	pulse on compare(TIMERx(x=0~4,7,19), TIMER_CH_x(x=2~3))
<i>TIMER_OC_MODE_DIR</i>	direction bit output(TIMERx(x=0~4,7,19), TIMER_CH_x(x=2~3))
<i>TIMER_OC_MODE_APWM0</i>	asymmetric PWM mode 0 (TIMERx(x=0~4,7,19))
<i>TIMER_OC_MODE_APWM1</i>	asymmetric PWM mode 0 (TIMERx(x=0~4,7,19))
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1182. Function timer\_channel\_output\_pulse\_value\_config**

<b>Function name</b>	timer_channel_output_pulse_value_config
<b>Function prototype</b>	void timer_channel_output_pulse_value_config(uint32_t timer_periph, uint16_t channel, uint32_t pulse);
<b>Function descriptions</b>	configure TIMER channel output pulse value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4, 7, 14~16, 19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4, 7, 14~16, 19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4, 7, 14, 19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4, 7, 19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4, 7, 19))
TIMER_MCH_0	TIMER multi mode channel 0 (TIMERx(x=0, 7, 14~16, 19))
TIMER_MCH_1	TIMER multi mode channel 1 (TIMERx(x=0, 7, 19))
TIMER_MCH_2	TIMER multi mode channel 2 (TIMERx(x=0, 7, 19))
TIMER_MCH_3	TIMER multi mode channel 3 (TIMERx(x=0, 7, 19))
<b>Input parameter{in}</b>	
<b>pulse</b>	the integer part of channel output pulse value, for TIMER_CH_x(x=0~3): 0~0xFFFF (adjustment mode disable, TIMERx(x=0, 2, 3, 5~7, 14~16, 19)) 0~0xFFFFE (adjustment mode enable, TIMERx(x=0, 2, 3, 5~7, 14~16, 19)) 0~0xFFFFFFFF (adjustment mode disable, TIMERx(x=1, 4)) 0~0xFFFFFFFFE (adjustment mode enable, TIMERx(x=1, 4)) for TIMER_MCH_x(x=0~3): 0~0xFFFF (TIMERx(x=0, 7, 14~16, 19))
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel 0 output pulse value */
```

```
timer_channel_output_pulse_value_config(TIMER0, TIMER_CH_0, 399);
```

## timer\_channel\_output\_pulse\_fract\_value\_config

The description of timer\_channel\_output\_pulse\_fract\_value\_config is shown as below:

**Table 3-1183. Function timer\_channel\_output\_pulse\_fract\_value\_config**

<b>Function name</b>	timer_channel_output_pulse_fract_value_config
<b>Function prototype</b>	void timer_channel_output_pulse_fract_value_config(uint32_t timer_periph, uint16_t channel, uint32_t fract);
<b>Function descriptions</b>	configure TIMER channel fractional part of output pulse value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4, 7, 14~16, 19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4, 7, 14~16, 19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4, 7, 14, 19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4, 7, 19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4, 7, 19))
<b>Input parameter{in}</b>	
<b>fract</b>	the fractional part of channel output pulse value, 0~15
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER channel fractional part of output pulse value */
timer_channel_output_pulse_fract_value_config (TIMER0, TIMER_CH_0, 15);
```

## timer\_channel\_output\_shadow\_config

The description of timer\_channel\_output\_shadow\_config is shown as below:

**Table 3-1184. Function timer\_channel\_output\_shadow\_config**

<b>Function name</b>	timer_channel_output_shadow_config
<b>Function prototype</b>	void timer_channel_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t ocshadow);
<b>Function descriptions</b>	configure TIMER channel output shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 ( <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19))
<i>TIMER_CH_1</i>	TIMER channel 1 ( <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2 ( <i>TIMERx</i> ( <i>x</i> =0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3 ( <i>TIMERx</i> ( <i>x</i> =0~4,7,19))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 ( <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 ( <i>TIMERx</i> ( <i>x</i> =0,7,19))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 ( <i>TIMERx</i> ( <i>x</i> =0,7,19))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 ( <i>TIMERx</i> ( <i>x</i> =0,7,19))
<b>Input parameter{in}</b>	
<b>ocshadow</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1185. Function timer\_channel\_output\_clear\_config**

<b>Function name</b>	timer_channel_output_clear_config
<b>Function prototype</b>	void timer_channel_output_clear_config(uint32_t timer_periph, uint16_t channel, uint16_t occlear);
<b>Function descriptions</b>	configure TIMER channel output clear function
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
timer_periph	TIMER periph
TIMERx(x=0~4,7,14~16,19)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4,7,19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4,7,19))
TIMER_MCH_0	TIMER multi mode channel 0 (TIMERx(x=0,7,14~16,19))
TIMER_MCH_1	TIMER multi mode channel 1 (TIMERx(x=0,7,19))
TIMER_MCH_2	TIMER multi mode channel 2 (TIMERx(x=0,7,19))
TIMER_MCH_3	TIMER multi mode channel 3 (TIMERx(x=0,7,19))
Input parameter{in}	
occlear	channel output clear function
TIMER_OC_CLEAR_ENABLE	channel output clear function enable
TIMER_OC_CLEAR_DISABLE	channel output clear function disable
TIMER_OMC_CLEAR_ENABLE	multi mode channel output clear function enable
TIMER_OMC_CLEAR_DISABLE	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\_compare\_fast\_config

The description of timer\_channel\_output\_compare\_fast\_config is shown as below:

**Table 3-1186. Function timer\_channel\_output\_compare\_fast\_config**

Function name	timer_channel_output_compare_fast_config
Function prototype	void timer_channel_output_compare_fast_config(uint32_t timer_periph, uint16_t channel, uint16_t ocfast);

<b>Function descriptions</b>	configure TIMER channel output compare fast function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER periphera
<i>TIMERx(x=0~4,7,14~16,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0~4,7,19))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7,14~16,19))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7,19))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7,19))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7,19))
<b>Input parameter{in}</b>	
<b>occlear</b>	channel output clear function
<i>TIMER_OC_FAST_ENABLE</i>	channel output compare fast function enable
<i>TIMER_OC_FAST_DISABLE</i>	channel output compare fast function disable
<i>TIMER_OMC_FAST_ENABLE</i>	multi mode channel output compare fast function enable
<i>TIMER_OMC_FAST_DISABLE</i>	multi mode channel output compare fast function disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel0 output compare fast function */
```

```
timer_channel_output_compare_fast_config (TIMER0, TIMER_CH_0,
TIMER_OC_FAST_ENABLE);
```

### timer\_channel\_output\_polarity\_config

The description of timer\_channel\_output\_polarity\_config is shown as below:

**Table 3-1187. Function timer\_channel\_output\_polarity\_config**

<b>Function name</b>	timer_channel_output_polarity_config
----------------------	--------------------------------------

<b>Function prototype</b>	void timer_channel_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocpolarity);
<b>Function descriptions</b>	configure TIMER channel output polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4, 7, 14~16, 19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4, 7, 14~16, 19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4, 7, 14, 19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4, 7, 19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4, 7, 19))
TIMER_MCH_0	TIMER multi mode channel 0 (TIMERx(x=0, 7, 14~16, 19))
TIMER_MCH_1	TIMER multi mode channel 1 (TIMERx(x=0, 7, 19))
TIMER_MCH_2	TIMER multi mode channel 2 (TIMERx(x=0, 7, 19))
TIMER_MCH_3	TIMER multi mode channel 3 (TIMERx(x=0, 7, 19))
<b>Input parameter{in}</b>	
<b>ocpolarity</b>	channel output polarity
TIMER_OC_POLARITY_HIGH	channel output polarity is high
TIMER_OC_POLARITY_LOW	channel output polarity is low
TIMER_OMC_POLARITY_HIGH	multi mode channel output polarity is high
TIMER_OMC_POLARITY_LOW	multi mode channel output polarity is low
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1188. Function timer\_channel\_complementary\_output\_polarity\_config

<b>Function name</b>	timer_channel_complementary_output_polarity_config
<b>Function prototype</b>	void timer_channel_complementary_output_polarity_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnpolarity);
<b>Function descriptions</b>	configure TIMER channel complementary output polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4,7,14~16,19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4,7,19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4,7,19))
<b>Input parameter{in}</b>	
<b>ocnpolarity</b>	channel complementary output polarity
TIMER_OCN_POLARITY_HIGH	channel complementary output polarity is high
TIMER_OCN_POLARITY_LOW	channel complementary output polarity is low
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1189. Function timer\_channel\_output\_state\_config

<b>Function name</b>	timer_channel_output_state_config
<b>Function prototype</b>	void timer_channel_output_state_config(uint32_t timer_periph, uint16_t channel, uint32_t state);
<b>Function descriptions</b>	configure TIMER channel enable state
<b>Precondition</b>	-



<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 ( <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19))
<i>TIMER_CH_1</i>	TIMER channel 1 ( <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2 ( <i>TIMERx</i> ( <i>x</i> =0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3 ( <i>TIMERx</i> ( <i>x</i> =0~4,7,19))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 ( <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 ( <i>TIMERx</i> ( <i>x</i> =0,7,19))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 ( <i>TIMERx</i> ( <i>x</i> =0,7,19))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 ( <i>TIMERx</i> ( <i>x</i> =0,7,19))
<b>Input parameter{in}</b>	
<b>state</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1190. Function timer\_channel\_complementary\_output\_state\_config**

<b>Function name</b>	timer_channel_complementary_output_state_config
<b>Function prototype</b>	void timer_channel_complementary_output_state_config(uint32_t timer_periph, uint16_t channel, uint16_t ocnstate);
<b>Function descriptions</b>	configure TIMER channel complementary output enable state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0~4,7,19))
<b>Input parameter{in}</b>	
<b>state</b>	TIMER channel complementary output enable state
<i>TIMER_CCXN_ENABLE</i>	channel complementary enable
<i>TIMER_CCXN_DISABLE</i>	channel complementary disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1191. Function timer\_channel\_input\_struct\_para\_init**

<b>Function name</b>	timer_channel_input_struct_para_init
<b>Function prototype</b>	void timer_channel_input_struct_para_init(timer_ic_parameter_struct* icpara);
<b>Function descriptions</b>	initialize the parameters of TIMER channel input parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>icpara</b>	TIMER channel input parameter struct, the structure members can refer to <a href="#">Structure timer_ic_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1192. 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}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14~16,19)</i>	please refer to the following parameters
Input parameter{in}	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0~4,7,19))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7,14~16,19))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7,19))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7,19))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7,19))
Input parameter{in}	
<b>icpara</b>	TIMER channel input parameter struct, the structure members can refer to <a href="#">Structure timer_ic_parameter_struct</a> .
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_icitpara.icselection = TIMER_IC_SELECTION_DIRECTTTI;
```

```
timer_icitpara.icprescaler = TIMER_IC_PSC_DIV1;
```

```
timer_icitpara.icfilter      = 0x0;
```

```
timer_input_capture_config(TIMER0, TIMER_CH_0, &timer_icitpara);
```

### timer\_channel\_input\_capture\_prescaler\_config

The description of timer\_channel\_input\_capture\_prescaler\_config is shown as below:

**Table 3-1193. 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}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4, 7, 14~16, 19)	please refer to the following parameters
Input parameter{in}	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 ( <i>TIMERx</i> ( <i>x</i> =0~4, 7, 14~16, 19))
<i>TIMER_CH_1</i>	TIMER channel 1 ( <i>TIMERx</i> ( <i>x</i> =0~4, 7, 14, 19))
<i>TIMER_CH_2</i>	TIMER channel 2 ( <i>TIMERx</i> ( <i>x</i> =0~4, 7, 19))
<i>TIMER_CH_3</i>	TIMER channel 3 ( <i>TIMERx</i> ( <i>x</i> =0~4, 7, 19))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 ( <i>TIMERx</i> ( <i>x</i> =0, 7, 14~16, 19))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 ( <i>TIMERx</i> ( <i>x</i> =0, 7, 19))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 ( <i>TIMERx</i> ( <i>x</i> =0, 7, 19))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 ( <i>TIMERx</i> ( <i>x</i> =0, 7, 19))
Input parameter{in}	
<b>prescaler</b>	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 3-1194. Function timer\_channel\_capture\_value\_register\_read**

Function name	timer_channel_capture_value_register_read
Function prototype	uint32_t timer_channel_capture_value_register_read(uint32_t timer_periph, uint16_t channel);
Function descriptions	read TIMER channel capture compare register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4, 7, 14~16, 19)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4, 7, 14~16, 19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4, 7, 14, 19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4, 7, 19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4, 7, 19))
TIMER_MCH_0	TIMER multi mode channel 0 (TIMERx(x=0, 7, 14~16, 19))
TIMER_MCH_1	TIMER multi mode channel 1 (TIMERx(x=0, 7, 19))
TIMER_MCH_2	TIMER multi mode channel 2 (TIMERx(x=0, 7, 19))
TIMER_MCH_3	TIMER multi mode channel 3 (TIMERx(x=0, 7, 19))
Output parameter{out}	
-	-
Return value	
uint32_t	channel capture compare register value (0~0xFFFFFFFF)

Example:

```
/* read TIMER0 channel 0 capture compare register value */
```

```
uint32_t ch0_value = 0;
```

```
ch0_value = timer_channel_capture_value_register_read(TIMER0, TIMER_CH_0);
```

### timer\_input\_pwm\_capture\_config

The description of timer\_input\_pwm\_capture\_config is shown as below:

Table 3-1195. Function timer\_input\_pwm\_capture\_config

Function name	timer_input_pwm_capture_config
Function prototype	void timer_input_pwm_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpwm);
Function descriptions	configure TIMER input pwm capture function
Precondition	-
The called functions	timer_channel_input_capture_prescaler_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14,19)	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0
TIMER_CH_1	TIMER channel 1
Input parameter{in}	
icpwm	TIMER channel input PWM parameter struct, the structure members can refer to <a href="#">Structure timer ic parameter struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 input pwm capture parameter */

timer_ic_parameter_struct timer_icinitpara;

timer_icinitpara.icpolarity = TIMER_IC_POLARITY_RISING;

timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTI;

timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;

timer_icinitpara.icfilter = 0x0;

timer_input_pwm_capture_config(TIMER0, TIMER_CH_0, &timer_icinitpara);

```

### timer\_hall\_mode\_config

The description of timer\_hall\_mode\_config is shown as below:

Table 3-1196. 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}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
Input parameter{in}	
<b>hallmode</b>	TIMER hall sensor mode state
<i>TIMER_HALLINTERFA CE_ENABLE</i>	TIMER hall sensor mode enable
<i>TIMER_HALLINTERFA CE_DISABLE</i>	TIMER hall sensor mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 hall sensor mode */
timer_hall_mode_config (TIMER0, TIMER_HALLINTERFACE_ENABLE);
```

### timer\_multi\_mode\_channel\_output\_parameter\_struct\_init

The description of timer\_multi\_mode\_channel\_output\_parameter\_struct\_init is shown as below:

**Table 3-1197. Function timer\_multi\_mode\_channel\_output\_parameter\_struct\_init**

<b>Function name</b>	timer_multi_mode_channel_output_parameter_struct_init
<b>Function prototype</b>	void timer_multi_mode_channel_output_parameter_struct_init(timer_omc_parameter_struct *omcpara);
<b>Function descriptions</b>	initialize TIMER multi mode channel output parameter struct
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>omcpara</b>	TIMER multi mode channel output parameter struct, the structure members can refer to <a href="#">Structure timer_omc_parameter_struct</a> .
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 3-1198. Function timer\_multi\_mode\_channel\_output\_config**

<b>Function name</b>	timer_multi_mode_channel_output_config
<b>Function prototype</b>	void timer_multi_mode_channel_output_config(uint32_t timer_periph, uint16_t channel, timer_omc_parameter_struct *omcpara);
<b>Function descriptions</b>	configure TIMER multi mode channel output function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7,14~16,19))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7,19))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7,19))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7,19))
<b>Input parameter{in}</b>	
<b>omcpara</b>	TIMER multi mode channel output parameter struct, the structure members can refer to <a href="#">Structure timer_omc_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1199. Function timer\_multi\_mode\_channel\_mode\_config**

<b>Function name</b>	timer_multi_mode_channel_mode_config
<b>Function prototype</b>	void timer_multi_mode_channel_mode_config(uint32_t timer_periph, uint32_t channel, uint32_t multi_mode_sel);
<b>Function descriptions</b>	multi mode channel mode select
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0,7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
TIMER_MCH_0	TIMER multi mode channel 0 (TIMERx(x=0,7,14~16,19))
TIMER_MCH_1	TIMER multi mode channel 1 (TIMERx(x=0,7,19))
TIMER_MCH_2	TIMER multi mode channel 2 (TIMERx(x=0,7,19))
TIMER_MCH_3	TIMER multi mode channel 3 (TIMERx(x=0,7,19))
<b>Input parameter{in}</b>	
<b>multi_mode_sel</b>	multi mode channel mode selection
TIMER_MCH_MODE_INDEPENDENTLY	multi mode channel work in independently mode
TIMER_MCH_MODE_COMPLEMENTARY	multi mode channel work in complementary output mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1200. Function timer\_input\_trigger\_source\_select**

<b>Function name</b>	timer_input_trigger_source_select
<b>Function prototype</b>	void timer_input_trigger_source_select(uint32_t timer_periph, uint32_t

	intrigger);
<b>Function descriptions</b>	select TIMER input trigger source
<b>Precondition</b>	SMC[2:0] = 000
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>intrigger</b>	input trigger source
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI0</i>	internal trigger input 0 (ITI0, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI1</i>	internal trigger input 1 (ITI1, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI2</i>	internal trigger input 2 (ITI2, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI3</i>	internal trigger input 3 (ITI3, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOF_ED</i>	TI0 edge detector (CIOF_ED, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOFE0</i>	filtered channel channel 0 input (CIOFE0, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI1FE1</i>	filtered channel channel 1 input (CI1FE1, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ETIFP</i>	external trigger input filter output (ETIFP, TIMERx(x=0~4,7,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI4</i>	internal trigger 4 (ITI4, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI5</i>	internal trigger 5 (ITI5, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI6</i>	internal trigger 6 (ITI6, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI7</i>	internal trigger 7 for (ITI7, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI8</i>	internal trigger 8 (ITI8, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI9</i>	internal trigger 9 (ITI9, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI10</i>	internal trigger 10 (ITI10, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI14</i>	internal trigger input 14 (ITI14, TIMERx (TIMERx(x=0~4,7,14,19))
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* select TIMER0 input trigger source */
```

```
timer_input_trigger_source_select (TIMER0, TIMER_SMCFG_TRGSEL_ITI0);
```

### timer\_master\_output0\_trigger\_source\_select

The description of timer\_master\_output0\_trigger\_source\_select is shown as below:

**Table 3-1201. Function timer\_master\_output0\_trigger\_source\_select**

Function name	timer_master_output0_trigger_source_select
Function prototype	void timer_master_output0_trigger_source_select(uint32_t timer_periph, uint32_t outrigger);
Function descriptions	select TIMER master mode output 0 trigger source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7,14,19)	TIMER peripheral selection
Input parameter{in}	
outrigger	trigger output source
TIMER_TRI_OUT0_SR_C_RESET	the UPG bit as trigger output 0 (TIMERx(x=0~7,14,19))
TIMER_TRI_OUT0_SR_C_ENABLE	the counter enable signal as trigger output 0 (TIMERx(x=0~7,14,19))
TIMER_TRI_OUT0_SR_C_UPDATE	update event as trigger output 0 (TIMERx(x=0~7,14,19))
TIMER_TRI_OUT0_SR_C_CH0	a capture or a compare match occurred in channel 0 as trigger output 0 (TIMERx(x=0~4,7,14,19))
TIMER_TRI_OUT0_SR_C_O0CPRE	O0CPRE as trigger output 0 (TIMERx(x=0~4,7,14,19))
TIMER_TRI_OUT0_SR_C_O1CPRE	O1CPRE as trigger output 0 (TIMERx(x=0~4,7,14,19))
TIMER_TRI_OUT0_SR_C_O2CPRE	O2CPRE as trigger output 0 (TIMERx(x=0~4,7,19))
TIMER_TRI_OUT0_SR_C_O3CPRE	O3CPRE as trigger output 0 (TIMERx(x=0~4,7,19))
TIMER_TRI_OUT0_SR_C_DECODER_CLOCK	decoder clock as trigger output 0 (TIMERx(x=0~4,7,19))
TIMER_TRI_OUT0_SR_C_SYNC	synchronization event as trigger output 0 (TIMERx(x=0,7,19))

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER0 master mode output 0 trigger source */
```

```
timer_master_output0_trigger_source_select (TIMER0, TIMER_TRI_OUT0_SRC_RESET);
```

### timer\_master\_output1\_trigger\_source\_select

The description of timer\_master\_output1\_trigger\_source\_select is shown as below:

**Table 3-1202. Function timer\_master\_output1\_trigger\_source\_select**

Function name	timer_master_output1_trigger_source_select
Function prototype	void timer_master_output1_trigger_source_select(uint32_t timer_periph, uint32_t outrigger);
Function descriptions	select TIMER master mode output 1 trigger source
Precondition	-
The called functions	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,19)</i>	TIMER peripheral selection
Input parameter{in}	
<b>outrigger</b>	trigger output source
<i>TIMER_TRI_OUT1_SRC_RESET</i>	the UPG bit as trigger output 1
<i>TIMER_TRI_OUT1_SRC_ENABLE</i>	the counter enable signal as trigger output 1
<i>TIMER_TRI_OUT1_SRC_UPDATE</i>	update event as trigger output 1
<i>TIMER_TRI_OUT1_SRC_CH0</i>	a capture or a compare match occurred in channel 0 as trigger output 1
<i>TIMER_TRI_OUT1_SRC_O0CPRE</i>	O0CPRE as trigger output 1
<i>TIMER_TRI_OUT1_SRC_O1CPRE</i>	O1CPRE as trigger output 1
<i>TIMER_TRI_OUT1_SRC_O2CPRE</i>	O2CPRE as trigger output 1
<i>TIMER_TRI_OUT1_SRC_O3CPRE</i>	O3CPRE as trigger output 1
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* select TIMER0 master mode output 1 trigger source */
```

```
timer_master_output1_trigger_source_select (TIMER0, TIMER_TRI_OUT1_SRC_RESET);
```

### timer\_slave\_mode\_select

The description of timer\_slave\_mode\_select is shown as below:

**Table 3-1203. Function timer\_slave\_mode\_select**

Function name	timer_slave_mode_select
Function prototype	void timer_slave_mode_select(uint32_t timer_periph, uint32_t slavemode);
Function descriptions	select TIMER slave mode
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14,19)	TIMER peripheral selection
Input parameter{in}	
slavemode	slave mode
TIMER_SLAVE_MODE_DISABLE	slave mode disable (TIMERx(x=0~4,7,14,19))
TIMER_ENCODER_MODE0	quadrature decoder mode 0 (TIMERx(x=0~4,7,19))
TIMER_ENCODER_MODE1	quadrature decoder mode 1 (TIMERx(x=0~4,7,19))
TIMER_ENCODER_MODE2	quadrature decoder mode 2 (TIMERx(x=0~4,7,19))
TIMER_SLAVE_MODE_RESTART	restart mode (TIMERx(x=0~4,7,14,19))
TIMER_SLAVE_MODE_PAUSE	pause mode (TIMERx(x=0~4,7,14,19))
TIMER_SLAVE_MODE_EVENT	event mode (TIMERx(x=0~4,7,14,19))
TIMER_SLAVE_MODE_EXTERNAL0	external clock mode 0 (TIMERx(x=0~4,7,14,19))
TIMER_SLAVE_MODE_RESTART_EVENT	restart + event mode (TIMERx(x=0~4,7,14,19))
TIMER_SLAVE_MODE_PAUSE_RESTART	pause + restart mode (TIMERx(x=0~4,7,14,19))
TIMER_DECODER_MODE0	decoder mode 0 (TIMERx(x=0~4,7,19))

<i>DE0</i>	
<i>TIMER_DECODER_MO DE1</i>	decoder mode 1 (TIMERx(x=0~4,7,19))
<i>TIMER_DECODER_MO DE2</i>	decoder mode 2 (TIMERx(x=0~4,7,19))
<i>TIMER_DECODER_MO DE3</i>	decoder mode 3 (TIMERx(x=0~4,7,19))
<i>TIMER_QUAD_DECOD ER_MODE3</i>	quadrature decoder mode 3 (TIMERx(x=0~4,7,19))
<i>TIMER_QUAD_DECOD ER_MODE4</i>	quadrature decoder mode 4 (TIMERx(x=0~4,7,19))
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0 slave mode */
```

```
timer_slave_mode_select (TIMER0, TIMER_ENCODER_MODE0);
```

### timer\_pause\_reset\_polarity\_config

The description of timer\_pause\_reset\_polarity\_config is shown as below:

**Table 3-1204. Function timer\_pause\_reset\_polarity\_config**

<b>Function name</b>	timer_pause_reset_polarity_config
<b>Function prototype</b>	void timer_pause_reset_polarity_config(uint32_t timer_periph, uint32_t rstpolarity);
<b>Function descriptions</b>	configure TIMER pause+reset mode reset polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>rstpolarity</b>	reset polarity
<i>TIMER_CNT_RESET_O N_FALLING_EDGE</i>	counter is reset on falling edge
<i>TIMER_CNT_RESET_O N_RISING_EDGE</i>	counter is reset on rising edge
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* configure TIMER pause+reset mode reset polarity */
timer_pause_reset_polarity_config (TIMER0, TIMER_CNT_RESET_ON_FALLING_EDGE);
```

### timer\_master\_slave\_mode\_config

The description of timer\_master\_slave\_mode\_config is shown as below:

**Table 3-1205. 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
TIMERx(x=0~4,7,14,19)	TIMER peripheral selection
Input parameter{in}	
masterslave	master slave mode state
TIMER_MASTER_SLAVE_MODE_ENABLE	master slave mode enable
TIMER_MASTER_SLAVE_MODE_DISABLE	master slave mode disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 master slave mode */
timer_master_slave_mode_config(TIMER0, TIMER_MASTER_SLAVE_MODE_ENABLE);
```

### timer\_external\_trigger\_config

The description of timer\_external\_trigger\_config is shown as below:

**Table 3-1206. 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

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>extprescaler</b>	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
<b>Input parameter{in}</b>	
<b>expolarity</b>	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
<b>Input parameter{in}</b>	
<b>extfilter</b>	external trigger filter control(0~15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1207. Function timer\_quadrature\_decoder\_mode\_config**

<b>Function name</b>	timer_quadrature_decoder_mode_config
<b>Function prototype</b>	void timer_quadrature_decoder_mode_config(uint32_t timer_periph, uint32_t decomode, uint16_t ic0polarity, uint16_t ic1polarity);
<b>Function descriptions</b>	configure TIMER quadrature decoder mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	



<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>decomode</b>	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
<i>TIMER_QUAD_DECODER_MODE3</i>	counter counts on CI0FE0 edge depending on CI0FE0 level
<i>TIMER_QUAD_DECODER_MODE4</i>	counter counts on CI1FE1 edge depending on CI1FE1 level
<b>Input parameter{in}</b>	
<b>ic0polarity</b>	CI0 input capture 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
<b>Input parameter{in}</b>	
<b>ic1polarity</b>	CI1 input capture 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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 quadrature decoder mode */
```

```
timer_quadrature_decoder_mode_config(TIMER0,          TIMER_ENCODER_MODE0,
timer_ic_polarity_rising, TIMER_IC_POLARITY_RISING);
```

### timer\_decoder\_mode\_config

The description of timer\_decoder\_mode\_config is shown as below:

Table 3-1208. Function timer\_decoder\_mode\_config

<b>Function name</b>	timer_decoder_mode_config
<b>Function prototype</b>	void timer_decoder_mode_config(uint32_t timer_periph, uint32_t decomode, uint16_t ic0polarity, uint16_t ic1polarity);
<b>Function descriptions</b>	configure TIMER decoder mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>decomode</b>	decoder mode
<i>TIMER_DECODER_MODE0</i>	decoder mode 0
<i>TIMER_DECODER_MODE1</i>	decoder mode 1
<i>TIMER_DECODER_MODE2</i>	decoder mode 2
<i>TIMER_DECODER_MODE3</i>	decoder mode 3
<b>Input parameter{in}</b>	
<b>ic0polarity</b>	CI0 input capture 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
<b>Input parameter{in}</b>	
<b>ic1polarity</b>	CI1 input capture 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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 decoder mode */
```

timer\_decoder\_mode\_config (TIMER0, TIMER\_DECODER\_MODE2,  
TIMER\_IC\_POLARITY\_RISING, TIMER\_IC\_POLARITY\_RISING);

### timer\_internal\_clock\_config

The description of timer\_internal\_clock\_config is shown as below:

**Table 3-1209. Function timer\_internal\_clock\_config**

<b>Function name</b>	timer_internal_clock_config
<b>Function prototype</b>	void timer_internal_clock_config(uint32_t timer_periph);
<b>Function descriptions</b>	configure TIMER internal clock mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,19)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1210. Function timer\_internal\_trigger\_as\_external\_clock\_config**

<b>Function name</b>	timer_internal_trigger_as_external_clock_config
<b>Function prototype</b>	void timer_internal_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t intrigger);
<b>Function descriptions</b>	configure TIMER the internal trigger as external clock input
<b>Precondition</b>	-
<b>The called functions</b>	timer_input_trigger_source_select
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>intrigger</b>	trigger selection
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITIO</i>	internal trigger input 0 (ITI0)
<i>TIMER_SMCFG_TRGS</i>	internal trigger input 1 (ITI1)

<i>EL_ITI1</i>	
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI2</i>	internal trigger input 2 (ITI2)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI3</i>	internal trigger input 3 (ITI3)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI4</i>	internal trigger 4 (ITI4)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI5</i>	internal trigger 5 (ITI5)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI6</i>	internal trigger 6 (ITI6)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI7</i>	internal trigger 7 for (ITI7)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI8</i>	internal trigger 8 (ITI8)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI9</i>	internal trigger 9 (ITI9)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI10</i>	internal trigger 10 (ITI10)
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI14</i>	internal trigger input 14 (ITI14)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1211. Function timer\_external\_trigger\_as\_external\_clock\_config**

<b>Function name</b>	timer_external_trigger_as_external_clock_config
<b>Function prototype</b>	void timer_external_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t extrigger, uint16_t expolarity, uint32_t extfilter);
<b>Function descriptions</b>	configure TIMER the external trigger as external clock input
<b>Precondition</b>	-
<b>The called functions</b>	timer_input_trigger_source_select
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral

<i>TIMERx(x=0~4,7,14,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>extrigger</b>	external trigger selection
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOF_ED</i>	TI0 edge detector (CIOF_ED, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOFE0</i>	filtered channel channel 0 input (CIOFE0, TIMERx(x=0~4,7,14,19))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI1FE1</i>	filtered channel channel 1 input (CI1FE1, TIMERx(x=0~4,7,14,19))
<b>Input parameter{in}</b>	
<b>expolarity</b>	external trigger polarity
<i>TIMER_IC_POLARITY_</i> <i>RISING</i>	active high or rising edge active
<i>TIMER_IC_POLARITY_</i> <i>FALLING</i>	active low or falling edge active
<i>TIMER_IC_POLARITY_</i> <i>BOTH_EDGE</i>	active both edge
<b>Input parameter{in}</b>	
<b>extfilter</b>	external trigger filter control(0~15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1212. Function timer\_external\_clock\_mode0\_config**

<b>Function name</b>	timer_external_clock_mode0_config
<b>Function prototype</b>	void timer_external_clock_mode0_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
<b>Function descriptions</b>	configure TIMER the external clock mode0
<b>Precondition</b>	-
<b>The called functions</b>	timer_external_trigger_config
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,19)</i>	TIMER peripheral selection

Input parameter{in}	
<b>extprescaler</b>	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}	
<b>expolarity</b>	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}	
<b>extfilter</b>	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 3-1213. Function timer\_external\_clock\_mode1\_config**

<b>Function name</b>	timer_external_clock_mode1_config
<b>Function prototype</b>	void timer_external_clock_mode1_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
<b>Function descriptions</b>	configure TIMER the external clock mode1
<b>Precondition</b>	-
<b>The called functions</b>	timer_external_trigger_config
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
Input parameter{in}	
<b>extprescaler</b>	ETI external trigger prescaler
<i>TIMER_EXT_TRI_PSC_</i>	no divided

OFF	
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
<b>Input parameter{in}</b>	
<b>expolarity</b>	ETI external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active
TIMER_ETP_RISING	active high or rising edge active
<b>Input parameter{in}</b>	
<b>extfilter</b>	ETI external trigger filter control(0~15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 the external clock mode1 */
```

```
timer_external_clock_mode1_config(TIMER0,          TIMER_EXT_TRI_PSC_DIV2,
timer_ETP_FALLING, 0);
```

### timer\_external\_clock\_mode1\_disable

The description of timer\_external\_clock\_mode1\_disable is shown as below:

**Table 3-1214. Function timer\_external\_clock\_mode1\_disable**

<b>Function name</b>	timer_external_clock_mode1_disable
<b>Function prototype</b>	void timer_external_clock_mode1_disable(uint32_t timer_periph);
<b>Function descriptions</b>	disable TIMER the external clock mode1
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4,7,19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TIMER0 the external clock mode1 */
```

```
timer_external_clock_mode1_disable(TIMER0);
```

### timer\_write\_chxval\_register\_config

The description of timer\_write\_chxval\_register\_config is shown as below:

**Table 3-1215. Function timer\_write\_chxval\_register\_config**

Function name	timer_write_chxval_register_config
Function prototype	void timer_write_chxval_register_config(uint32_t timer_periph, uint16_t ccsel);
Function descriptions	configure TIMER write CHxVAL register selection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4, 14~16, 19)	TIMER peripheral selection
Input parameter{in}	
ccsel	write CHxVAL register selection
TIMER_CHVSEL_DISABLE	no effect
TIMER_CHVSEL_ENABLE	when write the CHxVAL register, if the write value is same as the CHxVAL value, the write access is ignored
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 write CHxVAL register selection */
```

```
timer_write_chxval_register_config(TIMER0, TIMER_CHVSEL_ENABLE);
```

### timer\_output\_value\_selection\_config

The description of timer\_output\_value\_selection\_config is shown as below:

**Table 3-1216. 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}	



<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>outsel</b>	output value selection
<i>TIMER_OUTSEL_DISABLE</i>	no effect
<i>TIMER_OUTSEL_ENABLE</i>	if POEN and IOS is 0, the output disabled
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER output value selection */
```

```
timer_output_value_selection_config(TIMER0, TIMER_OUTSEL_ENABLE);
```

### timer\_commutation\_control\_shadow\_register\_config

The description of timer\_commutation\_control\_shadow\_register\_config is shown as below:

**Table 3-1217. Function timer\_commutation\_control\_shadow\_register\_config**

<b>Function name</b>	timer_commutation_control_shadow_register_config
<b>Function prototype</b>	void timer_commutation_control_shadow_register_config(uint32_t timer_periph, uint16_t ccssel);
<b>Function descriptions</b>	configure commutation control shadow register update selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>ccssel</b>	commutation control shadow register selection
<i>TIMER_CCUSEL_ENABLE</i>	the shadow registers update when the counter generates an overflow/underflow event
<i>TIMER_CCUSEL_DISABLE</i>	the shadow registers update when the counter generates an overflow/underflow event and the repetition counter value is zero
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure commutation control shadow register update selection */
```

```
timer_commutation_control_shadow_register_config (TIMER0, TIMER_CCUSEL_ENABLE);
```

### timer\_output\_match\_pulse\_select

The description of timer\_output\_match\_pulse\_select is shown as below:

**Table 3-1218. Function timer\_output\_match\_pulse\_select**

<b>Function name</b>	timer_output_match_pulse_select
<b>Function prototype</b>	void timer_output_match_pulse_select(uint32_t timer_periph, uint32_t channel, uint16_t pulsesel);
<b>Function descriptions</b>	configure TIMER output match pulse selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4,7,14,19))	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4,7,19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4,7,19))
<b>Input parameter{in}</b>	
<b>pulsesel</b>	output match pulse selection
TIMER_PULSE_OUTPUT T_NORMAL	channel output normal
TIMER_PULSE_OUTPUT T_CNT_UP	pulse output only when counting up
TIMER_PULSE_OUTPUT T_CNT_DOWN	pulse output only when counting down
TIMER_PULSE_OUTPUT T_CNT_BOTH	pulse output when counting up or down
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1219. Function timer\_channel\_composite\_pwm\_mode\_config**

<b>Function name</b>	timer_channel_composite_pwm_mode_config
<b>Function prototype</b>	void timer_channel_composite_pwm_mode_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure the TIMER composite PWM mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0~4,7,19))
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1220. Function timer\_channel\_composite\_pwm\_mode\_output\_pulse\_value\_config**

<b>Function name</b>	timer_channel_composite_pwm_mode_output_pulse_value_config
<b>Function prototype</b>	void

	timer_channel_composite_pwm_mode_output_pulse_value_config(uint32_t timer_periph, uint32_t channel, uint32_t pulse, uint32_t add_pulse);
<b>Function descriptions</b>	configure the TIMER composite PWM mode output pulse value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4, 7, 14, 19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0~4, 7, 14~16, 19))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0~4, 7, 14, 19))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0~4, 7, 19))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0~4, 7, 19))
<b>Input parameter{in}</b>	
<b>pulse</b>	channel compare value(0~0xFFFFFFFF)
<b>Input parameter{in}</b>	
<b>add_pulse</b>	channel additional compare value(0~0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1221. Function timer\_channel\_additional\_compare\_value\_config**

<b>Function name</b>	timer_channel_additional_compare_value_config
<b>Function prototype</b>	void timer_channel_additional_compare_value_config(uint32_t timer_periph, uint16_t channel, uint32_t value);
<b>Function descriptions</b>	configure TIMER channel additional compare value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4, 7, 14, 19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel

<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0~4,7,19))
<b>Input parameter{in}</b>	
<b>value</b>	channel additional compare value(0~0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1222. Function timer\_channel\_additional\_output\_shadow\_config**

<b>Function name</b>	timer_channel_additional_output_shadow_config
<b>Function prototype</b>	void timer_channel_additional_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t aocshadow);
<b>Function descriptions</b>	configure TIMER channel additional output shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14~16,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0~4,7,19))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0~4,7,19))
<b>Input parameter{in}</b>	
<b>aocshadow</b>	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
<b>Output parameter{out}</b>	

-	-
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\_channel\_additional\_output\_update\_select

The description of timer\_channel\_additional\_output\_update\_select is shown as below:

**Table 3-1223. Function timer\_channel\_additional\_output\_update\_select**

<b>Function name</b>	timer_channel_additional_output_update_select
<b>Function prototype</b>	void timer_channel_additional_output_update_select(uint32_t timer_periph, uint16_t channel, uint16_t update);
<b>Function descriptions</b>	select TIMER channel additional output register update source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4,7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4,7,14~16,19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4,7,14,19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4,7,19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4,7,19))
<b>Input parameter{in}</b>	
<b>update</b>	channel additional output register update
TIMER_ADD_UPDATE_BY_UPDATE	channel additional compare value register update by update event
TIMER_ADD_UPDATE_BY_COM_MATCH	channel additional compare value register update by compare value match event
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER channel additional output register update source */
```

timer\_channel\_additional\_output\_update\_select (TIMER0, TIMER\_CH\_0,  
TIMER\_ADD\_UPDATE\_BY\_UPDATE);

### timer\_channel\_additional\_compare\_value\_read

The description of timer\_channel\_additional\_compare\_value\_read is shown as below:

**Table 3-1224. Function timer\_channel\_additional\_compare\_value\_read**

Function name	timer_channel_additional_compare_value_read
Function prototype	uint32_t timer_channel_additional_compare_value_read(uint32_t timer_periph, uint16_t channel);
Function descriptions	read TIMER channel additional compare value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4, 7, 14~16, 19)	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0 (TIMERx(x=0~4, 7, 14~16, 19))
TIMER_CH_1	TIMER channel 1 (TIMERx(x=0~4, 7, 14, 19))
TIMER_CH_2	TIMER channel 2 (TIMERx(x=0~4, 7, 19))
TIMER_CH_3	TIMER channel 3 (TIMERx(x=0~4, 7, 19))
Output parameter{out}	
-	-
Return value	
uint32_t	channel additional compare value, 0~0xFFFFFFFF

Example:

```
/* get TIMER autoreload register value */
```

```
uint32_t i = 0;
```

```
i =timer_channel_additional_compare_value_read(TIMER0, TIMER_CH_0);
```

### timer\_break\_external\_source\_config

The description of timer\_break\_external\_source\_config is shown as below:

**Table 3-1225. Function timer\_break\_external\_source\_config**

Function name	timer_break_external_source_config
Function prototype	void timer_break_external_source_config(uint32_t timer_periph, uint16_t break_num, uint32_t break_src, ControlStatus newvalue);
Function descriptions	configure the TIMER break source
Precondition	-

The called functions	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> (x=0,7,14~16,19)	please refer to the following parameters
Input parameter{in}	
<b>break_num</b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	BREAK0 input signals, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BREAK1</i>	BREAK1 input signals, <i>TIMERx</i> (x=0,7,19)
Input parameter{in}	
<b>break_src</b>	break source
<i>TIMER_BRKIN0</i>	BRKIN0 alternate function input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKIN1</i>	BRKIN1 alternate function input enable, <i>TIMERx</i> (x=0,7,19)
<i>TIMER_BRKIN2</i>	BRKIN2 alternate function input enable, <i>TIMERx</i> (x=0,7,19)
<i>TIMER_BRKCOMP0</i>	CMP0 input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKCOMP1</i>	CMP1 input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKCOMP2</i>	CMP2 input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKCOMP3</i>	CMP3 input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKCOMP4</i>	CMP4 input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKCOMP5</i>	CMP5 input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKCOMP6</i>	CMP6 input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKCOMP7</i>	CMP7 input enable, <i>TIMERx</i> (x=0,7,14~16,19)
<i>TIMER_BRKHPDF</i>	HPDF input enable, <i>TIMERx</i> (x=0,7,14~16,19)
Input parameter{in}	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TIMER break source */
```

```
timer_break_external_source_config(TIMER0,    TIMER_BREAK0,    TIMER_BRKIN0,
ENABLE);
```

### timer\_break\_external\_polarity\_config

The description of timer\_break\_external\_polarity\_config is shown as below:

**Table 3-1226. Function timer\_break\_external\_polarity\_config**

Function name	timer_break_external_polarity_config
---------------	--------------------------------------



<b>Function prototype</b>	void timer_break_external_polarity_config(uint32_t timer_periph, uint16_t break_num, uint32_t break_src, uint16_t bkinpolarity);
<b>Function descriptions</b>	configure TIMER break polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>break_num</b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	BREAK0 input signals, TIMERx(x=0,7,14~16,19)
<i>TIMER_BREAK1</i>	BREAK1 input signals, TIMERx(x=0,7,19)
<b>Input parameter{in}</b>	
<b>break_src</b>	break source
<i>TIMER_BRKIN0</i>	BRKIN0 alternate function input enable, TIMERx(x=0,7,14~16,19)
<i>TIMER_BRKIN1</i>	BRKIN1 alternate function input enable, TIMERx(x=0,7,19)
<i>TIMER_BRKIN2</i>	BRKIN2 alternate function input enable, TIMERx(x=0,7,19)
<i>TIMER_BRKCOMP0</i>	CMP0 input enable, TIMERx(x=0,7,14~16,19)
<i>TIMER_BRKCOMP1</i>	CMP1 input enable, TIMERx(x=0,7,14~16,19)
<i>TIMER_BRKCOMP2</i>	CMP2 input enable, TIMERx(x=0,7,14~16,19)
<i>TIMER_BRKCOMP3</i>	CMP3 input enable, TIMERx(x=0,7,14~16,19)
<b>Input parameter{in}</b>	
<b>bkinpolarity</b>	break polarity
<i>TIMER_BRKIN_POLARITY_LOW</i>	active low
<i>TIMER_BRKIN_POLARITY_HIGH</i>	active high
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER break polarity */
```

```
timer_break_external_polarity_config(TIMER0,    TIMER_BREAK0,    TIMER_BRKIN0,
timer_break_external_polarity_config(TIMER0,    TIMER_BREAK0,    TIMER_BRKIN0,
TIMER_BRKIN_POLARITY_HIGH);
```

### timer\_break\_lock\_config

The description of timer\_break\_lock\_config is shown as below:

Table 3-1227. Function timer\_break\_lock\_config

Function name	timer_break_lock_config
Function prototype	void timer_break_lock_config(uint32_t timer_periph, uint16_t break_num, ControlStatus newvalue);
Function descriptions	configure TIMER break lock function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMEx(x=0,7,14~16,19)	TIMER peripheral selection
Input parameter{in}	
break_num	TIMER BREAKx
TIMER_BREAK0	enable BREAK0 lock function, TIMEx(x=0,7,14~16,19)
TIMER_BREAK1	enable BREAK1 lock function, TIMEx(x=0,7,19)
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER break lock function */
```

```
timer_break_lock_config(TIMER0, TIMER_BREAK0, ENABLE);
```

### timer\_break\_lock\_release\_config

The description of timer\_break\_lock\_release\_config is shown as below:

Table 3-1228. Function timer\_break\_lock\_release\_config

Function name	timer_break_lock_release_config
Function prototype	void timer_break_lock_release_config(uint32_t timer_periph, uint16_t break_num, ControlStatus newvalue);
Function descriptions	release the TIMER break lock function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMEx(x=0,7,14~16,19)	TIMER peripheral selection

Input parameter{in}	
<b>break_num</b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	release the BREAK0 lock function, TIMERx(x=0,7,14~16,19)
<i>TIMER_BREAK1</i>	release the BREAK1 lock function, TIMERx(x=0,7,19)
Input parameter{in}	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* release the TIMER break lock function */
```

```
timer_break_lock_release_config (TIMER0, TIMER_BREAK0, ENABLE);
```

### timer\_dead\_time\_falling\_edge\_config

The description of timer\_dead\_time\_falling\_edge\_config is shown as below:

**Table 3-1229. Function timer\_dead\_time\_falling\_edge\_config**

<b>Function name</b>	timer_dead_time_falling_edge_config
<b>Function prototype</b>	void timer_dead_time_falling_edge_config(uint32_t timer_periph, uint32_t deadtime);
<b>Function descriptions</b>	
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,19)</i>	TIMER peripheral selection
Input parameter{in}	
<b>deadtime</b>	dead time (0~255)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TIMER falling edge dead time */
```

```
timer_dead_time_falling_edge_config (TIMER0, 255);
```

## timer\_dead\_time\_different\_config

The description of timer\_dead\_time\_different\_config is shown as below:

**Table 3-1230. Function timer\_dead\_time\_different\_config**

<b>Function name</b>	timer_dead_time_different_config
<b>Function prototype</b>	void timer_dead_time_different_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure the TIMER dead time different function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER dead time different function */
```

```
timer_dead_time_different_config (TIMER0, ENABLE);
```

## timer\_dead\_time\_modify\_config

The description of timer\_dead\_time\_modify\_config is shown as below:

**Table 3-1231. Function timer\_dead\_time\_modify\_config**

<b>Function name</b>	timer_dead_time_modify_config
<b>Function prototype</b>	void timer_dead_time_modify_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	

<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER dead time modified on-the-fly function */
```

```
timer_dead_time_modify_config (TIMER0, ENABLE);
```

### timer\_channel\_break\_control\_config

The description of timer\_channel\_break\_control\_config is shown as below:

**Table 3-1232. Function timer\_channel\_break\_control\_config**

<b>Function name</b>	timer_channel_break_control_config
<b>Function prototype</b>	void timer_channel_break_control_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure the TIMER channel break function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	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
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1233. Function timer\_channel\_dead\_time\_config**

<b>Function name</b>	timer_channel_dead_time_config
<b>Function prototype</b>	void timer_channel_dead_time_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure the TIMER channel free dead time function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	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
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1234. Function timer\_free\_complementary\_struct\_para\_init**

<b>Function name</b>	timer_free_complementary_struct_para_init
<b>Function prototype</b>	void timer_free_complementary_struct_para_init(timer_free_complementary_parameter_struct *freecompara);

<b>Function descriptions</b>	initialize TIMER channel free complementary parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>freecompara</b>	TIMER channel free complementary parameter struct, the structure members can refer to <a href="#">Structure timer free complementary parameter struct.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1235. Function timer\_channel\_free\_complementary\_config**

<b>Function name</b>	timer_channel_free_complementary_config
<b>Function prototype</b>	void timer_channel_free_complementary_config(uint32_t timer_periph, uint16_t channel, timer_free_complementary_parameter_struct *fcpa);
<b>Function descriptions</b>	configure channel free complementary protection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	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
<b>Input parameter{in}</b>	
<b>freecompara</b>	TIMER channel free complementary parameter struct, the structure members can refer to <a href="#">Structure timer free complementary parameter struct.</a>
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```

/* initialize TIMER break parameter struct with a default value */

timer_free_complementary_parameter_struct timer_freecompara;

timer_freecompara.freecomstate = TIMER_FCCHP_STATE_ENABLE;

timer_freecompara.runoffstate   = TIMER_ROS_STATE_ENABLE;

timer_freecompara.ideloffstate  = TIMER_IOS_STATE_ENABLE;

timer_freecompara.deadtime      = 255;

timer_channel_free_complementary_config(&timer_freecompara);

```

### timer\_watchdog\_value\_config

The description of timer\_watchdog\_value\_config is shown as below:

**Table 3-1236. Function timer\_watchdog\_value\_config**

<b>Function name</b>	timer_watchdog_value_config
<b>Function prototype</b>	void timer_watchdog_value_config(uint32_t timer_periph, uint32_t value);
<b>Function descriptions</b>	configure TIMER autoreload register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>value</b>	watchdog counter period value, 0~0xFFFFFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure quadrature decoder signal disconnection detection watchdog value */

timer_watchdog_value_config(TIMER0, 3000);

```

### timer\_watchdog\_value\_read

The description of timer\_watchdog\_value\_read is shown as below:



Table 3-1237. Function timer\_watchdog\_value\_read

<b>Function name</b>	timer_watchdog_value_read
<b>Function prototype</b>	uint32_t timer_watchdog_value_read(uint32_t timer_periph);
<b>Function descriptions</b>	read quadrature decoder signal disconnection detection watchdog value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	watchdog counter period register value, 0~0xFFFFFFFF

Example:

```

/* read quadrature decoder signal disconnection detection watchdog value */
uint32_t i = 0;

i = timer_watchdog_value_read(TIMER0);

```

### timer\_decoder\_disconnection\_detection\_config

The description of timer\_decoder\_disconnection\_detection\_config is shown as below:

Table 3-1238. Function timer\_decoder\_disconnection\_detection\_config

<b>Function name</b>	timer_decoder_disconnection_detection_config
<b>Function prototype</b>	void timer_decoder_disconnection_detection_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure quadrature decoder signal disconnection detection function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure quadrature decoder signal disconnection detection function */
```

```
timer_decoder_disconnection_detection_config(TIMER0, ENABLE);
```

### timer\_decoder\_jump\_detection\_config

The description of timer\_decoder\_jump\_detection\_config is shown as below:

**Table 3-1239. Function timer\_decoder\_jump\_detection\_config**

<b>Function name</b>	timer_decoder_jump_detection_config
<b>Function prototype</b>	void timer_decoder_jump_detection_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure quadrature decoder signal jump detection function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure quadrature decoder signal jump detection function */
```

```
timer_decoder_jump_detection_config(TIMER0, ENABLE);
```

### timer\_decoder\_modify\_config

The description of timer\_decoder\_modify\_config is shown as below:

**Table 3-1240. Function timer\_decoder\_modify\_config**

<b>Function name</b>	timer_decoder_modify_config
<b>Function prototype</b>	void timer_decoder_modify_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure decoder mode modified on-the-fly function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral

<i>TIMERx</i> ( <i>x</i> =0~4,7,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure decoder mode modified on-the-fly function */
```

```
timer_decoder_modify_config (TIMER0, ENABLE);
```

### timer\_decoder\_mode\_update\_source\_config

The description of timer\_decoder\_mode\_update\_source\_config is shown as below:

**Table 3-1241. Function timer\_decoder\_mode\_update\_source\_config**

<b>Function name</b>	timer_decoder_mode_update_source_config
<b>Function prototype</b>	void timer_decoder_mode_update_source_config(uint32_t timer_periph, uint32_t source);
<b>Function descriptions</b>	configure decoder mode update source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>source</b>	update source
<i>TIMER_DEC_UPDATE_BY_UPDATE</i>	decoder mode is updated by the update event
<i>TIMER_DEC_UPDATE_BY_INDEX</i>	decoder mode is updated by the index event
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure decoder mode update source */
```

```
timer_decoder_mode_update_source_config (TIMER0, TIMER_DEC_UPDATE_BY_UPDATE);
```

## timer\_index\_reset\_counter\_config

The description of timer\_index\_reset\_counter\_config is shown as below:

**Table 3-1242. Function timer\_index\_reset\_counter\_config**

<b>Function name</b>	timer_index_reset_counter_config
<b>Function prototype</b>	void timer_index_reset_counter_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure index signal reset counter function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure index signal reset counter function */
timer_index_reset_counter_config (TIMER0, ENABLE);
```

## timer\_index\_reset\_direction\_config

The description of timer\_index\_reset\_direction\_config is shown as below:

**Table 3-1243. Function timer\_index\_reset\_direction\_config**

<b>Function name</b>	timer_index_reset_direction_config
<b>Function prototype</b>	void timer_index_reset_direction_config(uint32_t timer_periph, uint32_t direction);
<b>Function descriptions</b>	configure index signal reset counter direction
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>direction</b>	index signal reset counter direction
<i>TIMER_INDEX_RESET</i>	index signal resets the counter when it counts up and down

<code>_DIRECTION_BOTH</code>	
<code>TIMER_INDEX_RESET_DIRECTION_UP</code>	index signal resets the counter when it counts up
<code>TIMER_INDEX_RESET_DIRECTION_DOWN</code>	index signal resets the counter when it counts down
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure index signal reset counter direction */
```

```
timer_index_reset_direction_config (TIMER0, TIMER_INDEX_RESET_DIRECTION_BOTH);
```

### timer\_decoder\_index\_position\_config

The description of `timer_decoder_index_position_config` is shown as below:

**Table 3-1244. Function `timer_decoder_index_position_config`**

<b>Function name</b>	<code>timer_decoder_index_position_config</code>
<b>Function prototype</b>	<code>void timer_decoder_index_position_config(uint32_t timer_periph, uint32_t position);</code>
<b>Function descriptions</b>	configure index position
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<code>TIMERx(x=0~4,7,19)</code>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>position</b>	index position
<code>TIMER_INDEX_POSITION_AB_00</code>	when AB inputs are 00, index event will reset the counter (for quadrature decoder modes 0~4)
<code>TIMER_INDEX_POSITION_AB_01</code>	when AB inputs are 01, index event will reset the counter (for quadrature decoder modes 0~4)
<code>TIMER_INDEX_POSITION_AB_10</code>	when AB inputs are 10, index event will reset the counter (for quadrature decoder modes 0~4)
<code>TIMER_INDEX_POSITION_AB_11</code>	when AB inputs are 11, index event will reset the counter (for quadrature decoder modes 0~4)
<code>TIMER_INDEX_POSITION_LOW</code>	when clock is low, index event will reset the counter (for decoder modes 0~3)
<code>TIMER_INDEX_POSITION_HIGH</code>	when clock is high, index event will reset the counter (for decoder modes 0~3)
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure index position */
```

```
timer_decoder_index_position_config (TIMER0, TIMER_INDEX_POSITION_AB_01);
```

### timer\_first\_index\_reset\_counter\_config

The description of timer\_first\_index\_reset\_counter\_config is shown as below:

**Table 3-1245. Function timer\_first\_index\_reset\_counter\_config**

<b>Function name</b>	timer_first_index_reset_counter_config
<b>Function prototype</b>	void timer_first_index_reset_counter_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure only the first index signal reset counter function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure only the first index signal reset counter function */
```

```
timer_first_index_reset_counter_config (TIMER0, ENABLE);
```

### timer\_upif\_backup\_config

The description of timer\_upif\_backup\_config is shown as below:

**Table 3-1246. Function timer\_upif\_backup\_config**

<b>Function name</b>	timer_upif_backup_config
<b>Function prototype</b>	void timer_upif_backup_config(uint32_t timer_periph, ControlStatus newvalue);

<b>Function descriptions</b>	configure the UPIF bit backup function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the UPIF bit backup function */
```

```
timer_upif_backup_config(TIMER0, ENABLE);
```

### timer\_upifbu\_bit\_get

The description of timer\_upifbu\_bit\_get is shown as below:

**Table 3-1247. Function timer\_upifbu\_bit\_get**

<b>Function name</b>	timer_upifbu_bit_get
<b>Function prototype</b>	UPIFBUSStatus timer_upifbu_bit_get(uint32_t timer_periph);
<b>Function descriptions</b>	get the UPIFBU bit in the <i>TIMERx_CNT</i> register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 19)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>UPIFBUSStatus</b>	UPIFBU bit state
<i>VALID_SET</i>	the UPIFBU is valid and value is 1
<i>VALID_RESET</i>	the UPIFBU is valid and value is 0
<i>INVALID</i>	the UPIFBU is invalid

Example:

```
/* get the UPIFBU bit in the TIMEx_CNT register */
```

```
UPIFBUSStatus upstatus = INVALID;
```

```
upstatus = timer_upifbu_bit_get (TIMER0);
```

### timer\_counter\_initial\_register\_config

The description of timer\_counter\_initial\_register\_config is shown as below:

**Table 3-1248. Function timer\_counter\_initial\_register\_config**

<b>Function name</b>	timer_counter_initial_register_config
<b>Function prototype</b>	void timer_counter_initial_register_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure counter initial value register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMEx(x=0,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure counter initial value register */
```

```
timer_counter_initial_register_config (TIMER0, ENABLE);
```

### timer\_counter\_initial\_config

The description of timer\_counter\_initial\_config is shown as below:

**Table 3-1249. Function timer\_counter\_initial\_config**

<b>Function name</b>	timer_counter_initial_config
<b>Function prototype</b>	void timer_counter_initial_config(uint32_t timer_periph, uint32_t value, uint32_t direction);
<b>Function descriptions</b>	configure counter initial value and direction
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	



<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>value</b>	counter initial value, 0~0xFFFF
<b>Input parameter{in}</b>	
<b>direction</b>	counter initial direction
<i>TIMER_INITIAL_DIRECTION_DOWN</i>	counter count down when synchronizaton event occurs
<i>TIMER_INITIAL_DIRECTION_UP</i>	counter count up when synchronizaton event occurs
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure counter initial value and direction */
```

```
timer_counter_initial_config (TIMER0, TIMER_INITIAL_DIRECTION_UP);
```

### timer\_synchronization\_event\_generate

The description of timer\_synchronization\_event\_generate is shown as below:

**Table 3-1250. Function timer\_synchronization\_event\_generate**

<b>Function name</b>	timer_synchronization_event_generate
<b>Function prototype</b>	void timer_synchronization_event_generate(uint32_t timer_periph);
<b>Function descriptions</b>	generate soft synchronization event
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,19)</i>	please refer to the following parameters
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* generate soft synchronization event */
```

```
timer_synchronization_event_generate (TIMER0);
```

## timer\_flag\_get

The description of timer\_flag\_get is shown as below:

**Table 3-1251. Function timer\_flag\_get**

<b>Function name</b>	timer_flag_get
<b>Function prototype</b>	FlagStatus timer_flag_get(uint32_t timer_periph, uint32_t flag);
<b>Function descriptions</b>	get TIMER flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>flag</b>	the TIMER flags
TIMER_FLAG_UP	update flag, TIMERx(x=0~7,14~16,19)
TIMER_FLAG_CH0	channel 0 capture or compare flag, TIMERx(x=0~4,7,14~16,19)
TIMER_FLAG_CH1	channel 1 capture or compare flag, TIMERx(x=0~4,7,14,19)
TIMER_FLAG_CH2	channel 2 capture or compare flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_CH3	channel 3 capture or compare flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_CMT	channel commutation flag, TIMERx(x=0,7,14~16,19)
TIMER_FLAG_TRG	trigger flag, TIMERx(x=0~4,7,14,19)
TIMER_FLAG_BRK0	BREAK0 flag, TIMERx(x=0,7,14~16,19)
TIMER_FLAG_BRK1	BREAK1 flag, TIMERx(x=0,7,19)
TIMER_FLAG_CH0O	channel 0 overcapture flag, TIMERx(x=0~4,7,14~16,19)
TIMER_FLAG_CH1O	channel 1 overcapture flag, TIMERx(x=0~4,7,14,19)
TIMER_FLAG_CH2O	channel 2 overcapture flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_CH3O	channel 3 overcapture flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_SYSB	system source break flag, TIMERx(x=0,7,19)
TIMER_FLAG_IND	index flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_DECJ	quadrature decoder signal jump flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_DECDIS	quadrature decoder signal disconnection flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_DIRTRAN	direction transform flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_INDERR	index error flag, TIMERx(x=0~4,7,19)
TIMER_FLAG_MCH0	multi mode channel 0 capture or compare flag, TIMERx(x=0,7,14~16,19)
TIMER_FLAG_MCH1	multi mode channel 1 capture or compare flag, TIMERx(x=0,7,19)
TIMER_FLAG_MCH2	multi mode channel 2 capture or compare flag, TIMERx(x=0,7,19)
TIMER_FLAG_MCH3	multi mode channel 3 capture or compare flag, TIMERx(x=0,7,19)
TIMER_FLAG_MCH0O	multi mode channel 0 overcapture flag, TIMERx(x=0,7,14~16,19)
TIMER_FLAG_MCH1O	multi mode channel 1 overcapture flag, TIMERx(x=0,7,19)
TIMER_FLAG_MCH2O	multi mode channel 2 overcapture flag, TIMERx(x=0,7,19)

<i>TIMER_FLAG_MCH3O</i>	multi mode channel 3 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_CH0CO</i> <i>MADD</i>	channel 0 additional compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_FLAG_CH1CO</i> <i>MADD</i>	channel 1 additional compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_FLAG_CH2CO</i> <i>MADD</i>	channel 2 additional compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_CH3CO</i> <i>MADD</i>	channel 3 additional compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-1252. Function timer\_flag\_clear**

<b>Function name</b>	timer_flag_clear
<b>Function prototype</b>	void timer_flag_clear(uint32_t timer_periph, uint32_t flag);
<b>Function descriptions</b>	clear TIMER flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>flag</b>	the timer interrupt flags
<i>TIMER_FLAG_UP</i>	update flag, <i>TIMERx</i> ( <i>x</i> =0~7,14~16,19)
<i>TIMER_FLAG_CH0</i>	channel 0 capture or compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_FLAG_CH1</i>	channel 1 capture or compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_FLAG_CH2</i>	channel 2 capture or compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_CH3</i>	channel 3 capture or compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_CMT</i>	channel commutation flag, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_FLAG_TRG</i>	trigger flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)

<i>TIMER_FLAG_BRK0</i>	BREAK0 flag, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_FLAG_BRK1</i>	BREAK1 flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_CH0O</i>	channel 0 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_FLAG_CH1O</i>	channel 1 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_FLAG_CH2O</i>	channel 2 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_CH3O</i>	channel 3 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_SYSB</i>	system source break flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_IND</i>	index flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_DECJ</i>	quadrature decoder signal jump flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_DECDIS</i>	quadrature decoder signal disconnection flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_DIRTRAN</i>	direction transform flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_INDERR</i>	index error flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_MCH0</i>	multi mode channel 0 capture or compare flag, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_FLAG_MCH1</i>	multi mode channel 1 capture or compare flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_MCH2</i>	multi mode channel 2 capture or compare flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_MCH3</i>	multi mode channel 3 capture or compare flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_MCH0O</i>	multi mode channel 0 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_FLAG_MCH1O</i>	multi mode channel 1 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_MCH2O</i>	multi mode channel 2 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_MCH3O</i>	multi mode channel 3 overcapture flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_FLAG_CH0COMADD</i>	channel 0 additional compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_FLAG_CH1COMADD</i>	channel 1 additional compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_FLAG_CH2COMADD</i>	channel 2 additional compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_FLAG_CH3COMADD</i>	channel 3 additional compare flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1253. Function timer\_interrupt\_enable

<b>Function name</b>	timer_interrupt_enable
<b>Function prototype</b>	void timer_interrupt_enable(uint32_t timer_periph, uint32_t interrupt);
<b>Function descriptions</b>	enable the TIMER interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,19)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>interrupt</b>	timer interrupt enable source
<i>TIMER_INT_UP</i>	update interrupt enable, <i>TIMERx(x=0~7,14~16,19)</i>
<i>TIMER_INT_CH0</i>	channel 0 capture or compare interrupt enable, <i>TIMERx(x=0~4,7,14~16,19)</i>
<i>TIMER_INT_CH1</i>	channel 1 capture or compare interrupt enable, <i>TIMERx(x=0~4,7,14,19)</i>
<i>TIMER_INT_CH2</i>	channel 2 capture or compare interrupt enable, <i>TIMERx(x=0~4,7,19)</i>
<i>TIMER_INT_CH3</i>	channel 3 capture or compare interrupt enable, <i>TIMERx(x=0~4,7,19)</i>
<i>TIMER_INT_CMT</i>	commutation interrupt enable, <i>TIMERx(x=0,7,14~16,19)</i>
<i>TIMER_INT_TRG</i>	trigger interrupt enable, <i>TIMERx(x=0~4,7,14,19)</i>
<i>TIMER_INT_BRK</i>	break interrupt enable, <i>TIMERx(x=0,7,14~16,19)</i>
<i>TIMER_INT_IND</i>	index interrupt, <i>TIMERx(x=0~4,7,19)</i>
<i>TIMER_INT_DECJ</i>	quadrature decoder signal jump interrupt, <i>TIMERx(x=0~4,7,19)</i>
<i>TIMER_INT_DECDIS</i>	quadrature decoder signal disconnection interrupt, <i>TIMERx(x=0~4,7,19)</i>
<i>TIMER_INT_DIRTRAN</i>	direction transform interrupt, <i>TIMERx(x=0~4,7,19)</i>
<i>TIMER_INT_INDERR</i>	index error interrupt, <i>TIMERx(x=0~4,7,19)</i>
<i>TIMER_INT_MCH0</i>	multi mode channel 0 capture or compare interrupt, <i>TIMERx(x=0,7,14~16,19)</i>
<i>TIMER_INT_MCH1</i>	multi mode channel 1 capture or compare interrupt, <i>TIMERx(x=0,7,19)</i>
<i>TIMER_INT_MCH2</i>	multi mode channel 2 capture or compare interrupt, <i>TIMERx(x=0,7,19)</i>
<i>TIMER_INT_MCH3</i>	multi mode channel 3 capture or compare interrupt, <i>TIMERx(x=0,7,19)</i>
<i>TIMER_INT_CH0COMADD</i>	channel 0 additional compare interrupt, <i>TIMERx(x=0~4,7,14,19)</i>
<i>TIMER_INT_CH1COMADD</i>	channel 1 additional compare interrupt, <i>TIMERx(x=0~4,7,14,19)</i>
<i>TIMER_INT_CH2COMADD</i>	channel 2 additional compare interrupt, <i>TIMERx(x=0~4,7,19)</i>
<i>TIMER_INT_CH3COMADD</i>	channel 3 additional compare interrupt, <i>TIMERx(x=0~4,7,19)</i>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

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 3-1254. Function timer\_interrupt\_disable**

Function name	timer_interrupt_disable
Function prototype	void timer_interrupt_disable (uint32_t timer_periph, uint32_t interrupt);
Function descriptions	disable the TIMER interrupt
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 19)	please refer to the following parameters
Input parameter{in}	
interrupt	timer interrupt disable source
TIMER_INT_UP	update interrupt enable, TIMERx(x=0~7, 14~16, 19)
TIMER_INT_CH0	channel 0 capture or compare interrupt enable, TIMERx(x=0~4, 7, 14~16, 19)
TIMER_INT_CH1	channel 1 capture or compare interrupt enable, TIMERx(x=0~4, 7, 14, 19)
TIMER_INT_CH2	channel 2 capture or compare interrupt enable, TIMERx(x=0~4, 7, 19)
TIMER_INT_CH3	channel 3 capture or compare interrupt enable, TIMERx(x=0~4, 7, 19)
TIMER_INT_CMT	commutation interrupt enable, TIMERx(x=0, 7, 14~16, 19)
TIMER_INT_TRG	trigger interrupt enable, TIMERx(x=0~4, 7, 14, 19)
TIMER_INT_BRK	break interrupt enable, TIMERx(x=0, 7, 14~16, 19)
TIMER_INT_IND	index interrupt, TIMERx(x=0~4, 7, 19)
TIMER_INT_DECJ	quadrature decoder signal jump interrupt, TIMERx(x=0~4, 7, 19)
TIMER_INT_DECDIS	quadrature decoder signal disconnection interrupt, TIMERx(x=0~4, 7, 19)
TIMER_INT_DIRTRAN	direction transform interrupt, TIMERx(x=0~4, 7, 19)
TIMER_INT_INDERR	index error interrupt, TIMERx(x=0~4, 7, 19)
TIMER_INT_MCH0	multi mode channel 0 capture or compare interrupt, TIMERx(x=0, 7, 14~16, 19)
TIMER_INT_MCH1	multi mode channel 1 capture or compare interrupt, TIMERx(x=0, 7, 19)
TIMER_INT_MCH2	multi mode channel 2 capture or compare interrupt, TIMERx(x=0, 7, 19)
TIMER_INT_MCH3	multi mode channel 3 capture or compare interrupt, TIMERx(x=0, 7, 19)
TIMER_INT_CH0COMADD	channel 0 additional compare interrupt, TIMERx(x=0~4, 7, 14, 19)

<i>TIMER_INT_CH1COMA</i> <i>DD</i>	channel 1 additional compare interrupt, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_INT_CH2COMA</i> <i>DD</i>	channel 2 additional compare interrupt, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_CH3COMA</i> <i>DD</i>	channel 3 additional compare interrupt, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1255. Function `timer_interrupt_flag_get`**

<b>Function name</b>	<code>timer_interrupt_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus timer_interrupt_flag_get(uint32_t timer_periph, uint32_t int_flag);</code>
<b>Function descriptions</b>	get timer interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>int_flag</b>	the timer interrupt bits
<i>TIMER_INT_FLAG_UP</i>	update interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~7,14~16,19)
<i>TIMER_INT_FLAG_CH0</i>	channel 0 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,19)
<i>TIMER_INT_FLAG_CH1</i>	channel 1 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_INT_FLAG_CH2</i>	channel 2 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_CH3</i>	channel 3 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_CMT</i>	channel commutation interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_INT_FLAG_TRG</i>	trigger interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_INT_FLAG_BRK0</i>	BREAK0 interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_INT_FLAG_BRK1</i>	BREAK1 interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)

1	
<i>TIMER_INT_FLAG_SYS</i> <i>B</i>	system source break interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_INT_FLAG_IND</i>	index interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_DEC</i> <i>J</i>	quadrature decoder signal jump interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_DEC</i> <i>DIS</i>	quadrature decoder signal disconnection interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_DIR</i> <i>TRAN</i>	direction transform interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_IND</i> <i>ERR</i>	index error interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_MC</i> <i>H0</i>	multi mode channel 0 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,19)
<i>TIMER_INT_FLAG_MC</i> <i>H1</i>	multi mode channel 1 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_INT_FLAG_MC</i> <i>H2</i>	multi mode channel 2 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_INT_FLAG_MC</i> <i>H3</i>	multi mode channel 3 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,19)
<i>TIMER_INT_FLAG_CH0</i> <i>COMADD</i>	channel 0 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_INT_FLAG_CH1</i> <i>COMADD</i>	channel 1 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_INT_FLAG_CH2</i> <i>COMADD</i>	channel 2 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_CH3</i> <i>COMADD</i>	channel 3 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-1256. Function timer\_interrupt\_flag\_clear

<b>Function name</b>	timer_interrupt_flag_clear
<b>Function prototype</b>	void timer_interrupt_flag_clear(uint32_t timer_periph, uint32_t int_flag);
<b>Function descriptions</b>	clear TIMER interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~7,14~16,19)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>int_flag</b>	the timer interrupt bits
TIMER_INT_FLAG_UP	update interrupt flag, TIMERx(x=0~7,14~16,19)
TIMER_INT_FLAG_CH0	channel 0 capture or compare interrupt flag, TIMERx(x=0~4,7,14~16,19)
TIMER_INT_FLAG_CH1	channel 1 capture or compare interrupt flag, TIMERx(x=0~4,7,14,19)
TIMER_INT_FLAG_CH2	channel 2 capture or compare interrupt flag, TIMERx(x=0~4,7,19)
TIMER_INT_FLAG_CH3	channel 3 capture or compare interrupt flag, TIMERx(x=0~4,7,19)
TIMER_INT_FLAG_CMT	channel commutation interrupt flag, TIMERx(x=0,7,14~16,19)
TIMER_INT_FLAG_TRG	trigger interrupt flag, TIMERx(x=0~4,7,14,19)
TIMER_INT_FLAG_BRK0	BREAK0 interrupt flag, TIMERx(x=0,7,14~16,19)
TIMER_INT_FLAG_BRK1	BREAK1 interrupt flag, TIMERx(x=0,7,19)
TIMER_INT_FLAG_SYSB	system source break interrupt flag, TIMERx(x=0,7,19)
TIMER_INT_FLAG_IND	index interrupt flag, TIMERx(x=0~4,7,19)
TIMER_INT_FLAG_DECJ	quadrature decoder signal jump interrupt flag, TIMERx(x=0~4,7,19)
TIMER_INT_FLAG_DECDIS	quadrature decoder signal disconnection interrupt flag, TIMERx(x=0~4,7,19)
TIMER_INT_FLAG_DIRTRAN	direction transform interrupt flag, TIMERx(x=0~4,7,19)
TIMER_INT_FLAG_INDErr	index error interrupt flag, TIMERx(x=0~4,7,19)
TIMER_INT_FLAG_MCH0	multi mode channel 0 capture or compare interrupt flag, TIMERx(x=0,7,14~16,19)
TIMER_INT_FLAG_MCH1	multi mode channel 1 capture or compare interrupt flag, TIMERx(x=0,7,19)
TIMER_INT_FLAG_MCH2	multi mode channel 2 capture or compare interrupt flag, TIMERx(x=0,7,19)
TIMER_INT_FLAG_MCH3	multi mode channel 3 capture or compare interrupt flag, TIMERx(x=0,7,19)

<i>TIMER_INT_FLAG_CH0</i> <i>COMADD</i>	channel 0 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_INT_FLAG_CH1</i> <i>COMADD</i>	channel 1 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,19)
<i>TIMER_INT_FLAG_CH2</i> <i>COMADD</i>	channel 2 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<i>TIMER_INT_FLAG_CH3</i> <i>COMADD</i>	channel 3 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,19)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear TIMER0 update interrupt flag */
```

```
timer_interrupt_flag_clear(TIMER0, TIMER_INT_FLAG_UP);
```

## 3.31. TMU

The Trigonometric Math Unit (TMU) is a fully configurable block that execute common trigonometric and arithmetic operations. The TMU can reduce the burden of CPU. The TMU registers are listed in chapter [3.31.1](#), the TMU firmware functions are introduced in chapter [3.31.2](#).

### 3.31.1. Descriptions of Peripheral registers

TMU registers are listed in the table shown as below:

**Table 3-1257. TMU Registers**

Registers	Descriptions
TMU_CS	TMU control and status register
TMU_IDATA	TMU input data register
TMU_ODATA	TMU output data register

### 3.31.2. Descriptions of Peripheral functions

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

**Table 3-1258. TMU firmware function**

Function name	Function description
tmu_deinit	reset TMU registers
tmu_struct_para_init	initialize the parameters of TMU struct with the default values
tmu_init	initialize TMU

Function name	Function description
tmu_dma_read_enable	enable TMU DMA read request
tmu_dma_read_disable	disable TMU DMA read request
tmu_dma_write_enable	enable TMU DMA write request
tmu_dma_write_disable	disable TMU DMA write request
tmu_one_q31_write	write one data in q1.31 format
tmu_two_q31_write	write two data in q1.31 format
tmu_two_q15_write	write two data in q1.15 format
tmu_one_f32_write	write one data in floating point format
tmu_two_f32_write	write two data in floating point format
tmu_one_q31_read	read one data in q1.31 format
tmu_two_q31_read	read two data in q1.31 format
tmu_two_q15_read	read two data in q1.15 format
tmu_one_f32_read	read one data in floating point format
tmu_two_f32_read	read two data in floating point format
tmu_flag_get	get TMU flag
tmu_flag_clear	clear TMU flag
tmu_interrupt_enable	enable TMU interrupt
tmu_interrupt_disable	disable TMU interrupt
tmu_interrupt_flag_get	get TMU interrupt flag
tmu_interrupt_flag_clear	clear TMU interrupt flag

### Structure tmu\_parameter\_struct

**Table 3-1259. Structure tmu\_parameter\_struct**

Member name	Function description
mode	mode of TMU operation(TMU_MODE_COS, TMU_MODE_SIN, TMU_MODE_ATAN2, TMU_MODE_MODULUS, TMU_MODE_ATAN, TMU_MODE_COSH, TMU_MODE_SINH, TMU_MODE_ATANH, TMU_MODE_LN, TMU_MODE_SQRT)
iterations_number	number of iterations selection(TMU_ITERATION_STEPS_x(x=4,8,12,...24))
scale	scaling factor(TMU_SCALING_FACTOR_x(x=1,2,4,8,16,32,64,128))
output_floating	output data floating point format enable(TMU_OUTPUT_FLOAT_DISABLE, TMU_OUTPUT_FLOAT_ENABLE)
input_floating	input data floating point format enable(TMU_INPUT_FLOAT_DISABLE, TMU_INPUT_FLOAT_ENABLE)
dma_read	DMA request to read TMU_ODATA(TMU_READ_DMA_DISABLE, TMU_READ_DMA_ENABLE)
dma_write	DMA request to write TMU_IDATA(TMU_WRITE_DMA_DISABLE, TMU_WRITE_DMA_ENABLE)
read_times	times the TMU_ODATA needs to be read(TMU_READ_TIMES_1, TMU_READ_TIMES_2)
write_times	times the TMU_IDATA needs to be write(TMU_WRITE_TIMES_1,

Member name	Function description
	TMU_WRITE_TIMES_2)
output_width	width of output data(TMU_OUTPUT_WIDTH_32, TMU_OUTPUT_WIDTH_16)
input_width	width of input data(TMU_INPUT_WIDTH_32, TMU_INPUT_WIDTH_16)

### tmu\_deinit

The description of tmu\_deinit is shown as below:

**Table 3-1260. Function tmu\_deinit**

<b>Function name</b>	tmu_deinit
<b>Function prototype</b>	void tmu_deinit(void);
<b>Function descriptions</b>	reset the TMU registers
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset TMU */
tmu_deinit();
```

### tmu\_struct\_para\_init

The description of tmu\_struct\_para\_init is shown as below:

**Table 3-1261. Function tmu\_struct\_para\_init**

<b>Function name</b>	tmu_struct_para_init
<b>Function prototype</b>	void tmu_struct_para_init(timer_parameter_struct* initpara);
<b>Function descriptions</b>	initialize the parameters of TMU struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
init_struct	pointer to TMU init parameter struct, the structure members can refer to <a href="#">Structure tmu parameter struct.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize TMU init parameter struct with a default value */
```

```
tmu_parameter_struct tmu_initpara;
```

```
tmu_struct_para_init(&tmu_initpara);
```

## tmu\_init

The description of tmu\_init is shown as below:

**Table 3-1262. Function tmu\_init**

<b>Function name</b>	tmu_init
<b>Function prototype</b>	void tmu_init(tmu_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize TMU
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>init_struct</b>	pointer to TMU init parameter struct, the structure members can refer to <a href="#">Structure tmu_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize TMU */
```

```
tmu_parameter_struct tmu_init_struct;
```

```
tmu_init_struct.mode = TMU_MODE_COS;
```

```
tmu_init_struct.iterations_number = TMU_ITERATION_STEPS_24;
```

```
tmu_init_struct.scale = TMU_SCALING_FACTOR_1;
```

```
tmu_init_struct.output_floating = TMU_OUTPUT_FLOAT_DISABLE;
```

```
tmu_init_struct.input_floating = TMU_INPUT_FLOAT_DISABLE;
```

```
tmu_init_struct.dma_read = TMU_READ_DMA_ENABLE;
```

```
tmu_init_struct.dma_write = TMU_WRITE_DMA_ENABLE;
```

```
tmu_init_struct.read_times = TMU_READ_TIMES_2;
```

```
tmu_init_struct.write_times = TMU_WRITE_TIMES_2;
```

```
tmu_init_struct.output_width = TMU_OUTPUT_WIDTH_32;
```

```
tmu_init_struct.input_width = TMU_INPUT_WIDTH_32;
```

```
tmu_init(&tmu_init_struct);
```

## tmu\_dma\_read\_enable

The description of tmu\_dma\_read\_enable is shown as below:

**Table 3-1263. Function tmu\_dma\_read\_enable**

<b>Function name</b>	tmu_dma_read_enable
<b>Function prototype</b>	void tmu_dma_read_enable(void);
<b>Function descriptions</b>	enable TMU read interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TMU DMA read request */
tmu_dma_read_enable();
```

## tmu\_dma\_read\_disable

The description of tmu\_dma\_read\_disable is shown as below:

**Table 3-1264. Function tmu\_dma\_read\_disable**

<b>Function name</b>	tmu_dma_read_disable
<b>Function prototype</b>	void tmu_dma_read_disable(void);
<b>Function descriptions</b>	disable TMU read interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TMU DMA read request */
tmu_dma_read_disable();
```

## tmu\_dma\_write\_enable

The description of tmu\_dma\_write\_enable is shown as below:

**Table 3-1265. Function tmu\_dma\_write\_enable**

<b>Function name</b>	tmu_dma_write_enable
<b>Function prototype</b>	void tmu_dma_write_enable(void);
<b>Function descriptions</b>	enable TMU write interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TMU DMA write request */
tmu_dma_write_enable();
```

## tmu\_dma\_write\_disable

The description of tmu\_dma\_write\_disable is shown as below:

**Table 3-1266. Function tmu\_dma\_write\_disable**

<b>Function name</b>	tmu_dma_write_disable
<b>Function prototype</b>	void tmu_dma_write_disable(void);
<b>Function descriptions</b>	disable TMU write interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TMU DMA write request */
tmu_dma_write_disable();
```

## tmu\_one\_q31\_write

The description of tmu\_one\_q31\_write is shown as below:

**Table 3-1267. Function tmu\_one\_q31\_write**

<b>Function name</b>	tmu_one_q31_write
<b>Function prototype</b>	void tmu_one_q31_write(uint32_t data);
<b>Function descriptions</b>	write one data in q1.31 format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data</b>	the input data in q1.31 format (0x00000000~0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write onedata in q1.31 format */
```

```
int32_t in = -2000;
```

```
tmu_one_q31_write((uint32_t)in);
```

## tmu\_two\_q31\_write

The description of tmu\_two\_q31\_write is shown as below:

**Table 3-1268. Function tmu\_two\_q31\_write**

<b>Function name</b>	tmu_two_q31_write
<b>Function prototype</b>	void tmu_two_q31_write(uint32_t data1, uint32_t data2);
<b>Function descriptions</b>	write two data in q1.31 format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data1</b>	the first input data in q1.31 format (0x00000000~0xFFFFFFFF)
<b>Input parameter{in}</b>	
<b>data2</b>	the second input data in q1.31 format (0x00000000~0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write two data in q1.31 format */
```



```
int32_t in1 = -2000;

int32_t in2 = 3000;

tmu_two_q31_write((uint32_t)in1, (uint32_t)in2);
```

### tmu\_two\_q15\_write

The description of tmu\_two\_q15\_write is shown as below:

**Table 3-1269. Function tmu\_two\_q15\_write**

<b>Function name</b>	tmu_two_q15_write
<b>Function prototype</b>	void tmu_two_q15_write(uint16_t data1, uint16_t data2);
<b>Function descriptions</b>	write two data in q1.15 format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data1</b>	the first input data in q1.15 format (0x0000~0xFFFF)
<b>Input parameter{in}</b>	
<b>data2</b>	the second input data in q1.15 format (0x0000~0xFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write two data in q1.15 format */

int16_t in1 = -2000;

int16_t in2 = 3000;

tmu_two_q15_write((uint16_t)in1, (uint16_t)in2);
```

### tmu\_one\_f32\_write

The description of tmu\_one\_f32\_write is shown as below:

**Table 3-1270. Function tmu\_one\_f32\_write**

<b>Function name</b>	tmu_one_f32_write
<b>Function prototype</b>	void tmu_one_f32_write(float data);
<b>Function descriptions</b>	write one data in floating point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data</b>	the input data in floating point format
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* write one data in floating point format */
```

```
float in = -2000.0f;
```

```
tmu_one_f32_write(in);
```

### tmu\_two\_f32\_write

The description of tmu\_two\_f32\_write is shown as below:

**Table 3-1271. Function tmu\_two\_f32\_write**

<b>Function name</b>	tmu_two_f32_write
<b>Function prototype</b>	void tmu_two_f32_write(float data1, float data2);
<b>Function descriptions</b>	write two data in floating point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data1</b>	the first input data in floating point format
<b>Input parameter{in}</b>	
<b>data2</b>	the second input data in floating point format
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write two data in floating point format */
```

```
float in1 = -2000.0f;
```

```
float in2 = 3000.0f;
```

```
tmu_two_f32_write(in1, in2);
```

### tmu\_one\_q31\_read

The description of tmu\_one\_q31\_read is shown as below:

**Table 3-1272. Function tmu\_one\_q31\_read**

<b>Function name</b>	tmu_one_q31_read
<b>Function prototype</b>	void tmu_one_q31_read(uint32_t* p);
<b>Function descriptions</b>	read one data in q1.31 format

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>p</b>	A pointer to output data(q1.31 format)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* read one data in q1.31 format */
```

```
uint32_t out = 0;
```

```
tmu_one_q31_read (&out);
```

### tmu\_two\_q31\_read

The description of tmu\_two\_q31\_read is shown as below:

**Table 3-1273. Function tmu\_two\_q31\_read**

<b>Function name</b>	tmu_two_q31_read
<b>Function prototype</b>	void tmu_two_q31_read(uint32_t* p1, uint32_t* p2);
<b>Function descriptions</b>	read two data in q1.31 format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>p1</b>	A pointer to the first output data(q1.31 format)
<b>Input parameter{in}</b>	
<b>p2</b>	A pointer to the second output data(q1.31 format)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* read two data in q1.31 format */
```

```
uint32_t out1 = 0;
```

```
uint32_t out2 = 0;
```

```
tmu_two_q31_read(&out1, &out2);
```

### tmu\_two\_q15\_read

The description of tmu\_two\_q15\_read is shown as below:

Table 3-1274. Function tmu\_two\_q15\_read

Function name	tmu_two_q15_read
Function prototype	void tmu_two_q15_read(uint16_t* p1, uint16_t* p2);
Function descriptions	read two data in q1.15 format
Precondition	-
The called functions	-
Input parameter{in}	
p1	A pointer to the first output data(q1.15 format)
Input parameter{in}	
p2	A pointer to the second output data(q1.15 format)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read two data in q1.15 format */

uint16_t out1 = 0;

uint16_t out2 = 0;

tmu_two_q15_read(&out1, &out2);
```

### tmu\_one\_f32\_read

The description of tmu\_one\_f32\_read is shown as below:

Table 3-1275. Function tmu\_one\_f32\_read

Function name	tmu_one_f32_read
Function prototype	void tmu_one_f32_read(float* p);
Function descriptions	read one data in floating point format
Precondition	-
The called functions	-
Input parameter{in}	
p	A pointer to output data(floating point format)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read one data in floating point format */

float out = 0.0f;
```

```
tmu_one_f32_read (&out);
```

## tmu\_two\_f32\_read

The description of tmu\_two\_f32\_read is shown as below:

**Table 3-1276. Function tmu\_two\_f32\_read**

Function name	tmu_two_f32_read
Function prototype	void tmu_two_f32_read(float* p1, float* p2)
Function descriptions	read two data in floating point format
Precondition	-
The called functions	-
Input parameter{in}	
<b>p1</b>	A pointer to the first output data(floating point format)
Input parameter{in}	
<b>p2</b>	A pointer to the second output data(floating point format)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read two data in floating point format */
```

```
float out1 = 0.0f;
```

```
float out2 = 0.0f;
```

```
tmu_two_f32_read(&out1, &out2);
```

## tmu\_flag\_get

The description of tmu\_flag\_get is shown as below:

**Table 3-1277. Function tmu\_flag\_get**

Function name	tmu_flag_get
Function prototype	FlagStatus tmu_flag_get(uint32_t flag);
Function descriptions	get TMU flag
Precondition	-
The called functions	-
Input parameter{in}	
<b>flag</b>	the TMU flags
<i>TMU_FLAG_OVRF</i>	TMU overflow error flag
<i>TMU_FLAG_END</i>	end of TMU operation flag
Output parameter{out}	
-	-

Return value	
FlagStatus	SET or RESET

Example:

```
/* get TMU flag bit */
```

```
FlagStatus flag_value;
```

```
flag_value = tmu_flag_get(TMU_FLAG_OVRF);
```

### tmu\_flag\_clear

The description of tmu\_flag\_clear is shown as below:

**Table 3-1278. Function tmu\_flag\_clear**

Function name	tmu_flag_clear
Function prototype	void tmu_flag_clear(uint32_t flag);
Function descriptions	clear TMU flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	the TMU flags
TMU_FLAG_OVRF	TMU overflow error flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear TMU flag bit */
```

```
tmu_flag_clear(TMU_FLAG_OVRF);
```

### tmu\_interrupt\_enable

The description of tmu\_interrupt\_enable is shown as below:

**Table 3-1279. Function tmu\_interrupt\_enable**

Function name	tmu_interrupt_enable
Function prototype	void tmu_interrupt_enable(uint32_t interrupt);
Function descriptions	enable TMU interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	the TMU interrupt
TMU_INT_OVRF	TMU overflow interrupt

<i>TMU_INT_END</i>	TMU request to read TMU_ODATA interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TMU interrupt */
tmu_interrupt_enable(TMU_INT_OVRF);
```

### tmu\_interrupt\_disable

The description of tmu\_interrupt\_disable is shown as below:

**Table 3-1280. Function tmu\_interrupt\_disable**

<b>Function name</b>	tmu_interrupt_disable
<b>Function prototype</b>	void tmu_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable TMU interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	the TMU interrupt
<i>TMU_INT_OVRF</i>	TMU overflow interrupt
<i>TMU_INT_END</i>	TMU request to read TMU_ODATA interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TMU interrupt */
tmu_interrupt_disable(TMU_INT_OVRF);
```

### tmu\_interrupt\_flag\_get

The description of tmu\_interrupt\_flag\_get is shown as below:

**Table 3-1281. Function tmu\_interrupt\_flag\_get**

<b>Function name</b>	tmu_interrupt_flag_get
<b>Function prototype</b>	FlagStatus tmu_interrupt_flag_get(uint32_t int_flag);
<b>Function descriptions</b>	get TMU interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>int_flag</b>	the TMU interrupt flags
<i>TMU_INT_FLAG_OVR</i> <i>F</i>	TMU overflow interrupt flag
<i>TMU_INT_FLAG_END</i>	TMU request to read TMU_ODATA interrupt flag
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the TMU interrupt flag bit*/
```

```
FlagStatus flag_value;
```

```
flag_value = tmu_interrupt_flag_get(TMU_INT_FLAG_OVRF);
```

### tmu\_interrupt\_flag\_clear

The description of tmu\_interrupt\_flag\_clear is shown as below:

**Table 3-1282. Function tmu\_interrupt\_flag\_clear**

<b>Function name</b>	tmu_interrupt_flag_clear
<b>Function prototype</b>	void tmu_interrupt_flag_clear(uint32_t int_flag);
<b>Function descriptions</b>	clear TMU interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>int_flag</b>	the TMU interrupt flags
<i>TMU_INT_FLAG_OVR</i> <i>F</i>	TMU overflow interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the TMU interrupt flag bit*/
```

```
tmu_interrupt_flag_clear(TMU_INT_FLAG_OVRF);
```

## 3.32. 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 [3.32.1](#), the



TRIGSEL firmware functions are introduced in chapter [3.32.2](#).

## 3.32.1. Descriptions of Peripheral registers

TRIGSEL registers are listed in the table shown as below:

**Table 3-1283. TRIGSEL Registers**

Registers	Descriptions
TRIGSEL_EXTOUT_0	TRIGSEL trigger selection for EXTOUT register 0
TRIGSEL_EXTOUT_1	TRIGSEL trigger selection for EXTOUT register 1
TRIGSEL_EXTOUT_2	TRIGSEL trigger selection for EXTOUT register 2
TRIGSEL_EXTOUT_3	TRIGSEL trigger selection for EXTOUT register 3
TRIGSEL_ADC0	TRIGSEL trigger selection for ADC0 register
TRIGSEL_ADC1	TRIGSEL trigger selection for ADC1 register
TRIGSEL_ADC2	TRIGSEL trigger selection for ADC2 register
TRIGSEL_ADC3	TRIGSEL trigger selection for ADC3 register
TRIGSEL_TIMER0BRKIN	TRIGSEL trigger selection for TIMER0_BRKIN register
TRIGSEL_TIMER7BRKIN	TRIGSEL trigger selection for TIMER7_BRKIN register
TRIGSEL_TIMER14BRKIN	TRIGSEL trigger selection for TIMER14_BRKIN register
TRIGSEL_TIMER15BRKIN	TRIGSEL trigger selection for TIMER15_BRKIN register
TRIGSEL_TIMER16BRKIN	TRIGSEL trigger selection for TIMER16_BRKIN register
TRIGSEL_TIMER19BRKIN	TRIGSEL trigger selection for TIMER19_BRKIN register
TRIGSEL_CAN0	TRIGSEL trigger selection for CAN0 register
TRIGSEL_CAN1	TRIGSEL trigger selection for CAN1 register
TRIGSEL_CAN2	TRIGSEL trigger selection for CAN2 register
TRIGSEL_TIMER0ETI	TRIGSEL trigger selection for TIMER0_ETI register
TRIGSEL_TIMER1ETI	TRIGSEL trigger selection for TIMER1_ETI register
TRIGSEL_TIMER2ETI	TRIGSEL trigger selection for TIMER2_ETI register
TRIGSEL_TIMER3ETI	TRIGSEL trigger selection for TIMER3_ETI register
TRIGSEL_TIMER4ETI	TRIGSEL trigger selection for TIMER4_ETI register
TRIGSEL_TIMER7ETI	TRIGSEL trigger selection for TIMER7_ETI register
TRIGSEL_TIMER19ETI	TRIGSEL trigger selection for TIMER19_ETI register
TRIGSEL_HPDPF	TRIGSEL trigger selection for HPDPF register
TRIGSEL_TIMER0ITI14	TRIGSEL trigger selection for TIMER0_ITI14 register
TRIGSEL_TIMER1ITI14	TRIGSEL trigger selection for TIMER1_ITI14 register
TRIGSEL_TIMER2ITI14	TRIGSEL trigger selection for TIMER2_ITI14 register
TRIGSEL_TIMER3ITI14	TRIGSEL trigger selection for TIMER3_ITI14 register
TRIGSEL_TIMER4ITI14	TRIGSEL trigger selection for TIMER4_ITI14 register
TRIGSEL_TIMER7ITI14	TRIGSEL trigger selection for TIMER7_ITI14 register
TRIGSEL_TIMER14ITI14	TRIGSEL trigger selection for TIMER14_ITI14 register
TRIGSEL_TIMER19ITI14	TRIGSEL trigger selection for TIMER19_ITI14 register
TRIGSEL_DAC0	TRIGSEL trigger selection for DAC0 register
TRIGSEL_DAC1	TRIGSEL trigger selection for DAC1 register

Registers	Descriptions
TRIGSEL_DAC2	TRIGSEL trigger selection for DAC2 register
TRIGSEL_DAC3	TRIGSEL trigger selection for DAC3 register
TRIGSEL_EXTDAC0	TRIGSEL trigger selection for DAC0 extended register
TRIGSEL_EXTDAC1	TRIGSEL trigger selection for DAC1 extended register
TRIGSEL_EXTDAC2	TRIGSEL trigger selection for DAC2 extended register
TRIGSEL_EXTDAC3	TRIGSEL trigger selection for DAC3 extended register
TRIGSEL_CLA_0	TRIGSEL trigger selection for CLA register 0
TRIGSEL_CLA_1	TRIGSEL trigger selection for CLA register 1
TRIGSEL_CLA_2	TRIGSEL trigger selection for CLA register 2
TRIGSEL_CLA_3	TRIGSEL trigger selection for CLA register 3
TRIGSEL_CLA_4	TRIGSEL trigger selection for CLA register 4

### 3.32.2. Descriptions of Peripheral functions

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

**Table 3-1284. TRIGSEL firmware function**

Function name	Function description
trigsel_deinit	deinitialize TRIGSEL
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 3-1285. 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

Member name	Function description
TRIGSEL_INPUT_TRIGSEL_IN12	trigger input source TRIGSEL_IN12 pin
TRIGSEL_INPUT_TRIGSEL_IN13	trigger input source TRIGSEL_IN13 pin
TRIGSEL_INPUT_TIMER0_TRGO0	trigger input source TIMER0 TRGO0
TRIGSEL_INPUT_TIMER0_TRGO1	trigger input source TIMER0 TRGO1
TRIGSEL_INPUT_TIMER0_CH0	trigger input source TIMER0 CH0
TRIGSEL_INPUT_TIMER0_CH1	trigger input source TIMER0 CH1
TRIGSEL_INPUT_TIMER0_CH2	trigger input source TIMER0 CH2
TRIGSEL_INPUT_TIMER0_CH3	trigger input source TIMER0 CH3
TRIGSEL_INPUT_TIMER0_MCH0	trigger input source TIMER0 MCH0
TRIGSEL_INPUT_TIMER0_MCH1	trigger input source TIMER0 MCH1
TRIGSEL_INPUT_TIMER0_MCH2	trigger input source TIMER0 MCH2
TRIGSEL_INPUT_TIMER0_MCH3	trigger input source TIMER0 MCH3
TRIGSEL_INPUT_TIMER0_ETI	trigger input source TIMER0 ETI
TRIGSEL_INPUT_TIMER1_TRGO0	trigger input source TIMER1 TRGO0
TRIGSEL_INPUT_TIMER1_CH0	trigger input source TIMER1 CH0
TRIGSEL_INPUT_TIMER1_CH1	trigger input source TIMER1 CH1
TRIGSEL_INPUT_TIMER1_CH2	trigger input source TIMER1 CH2
TRIGSEL_INPUT_TIMER1_CH3	trigger input source TIMER1 CH3
TRIGSEL_INPUT_TIMER1_ETI	trigger input source TIMER1 ETI
TRIGSEL_INPUT_TIMER2_TRGO0	trigger input source TIMER2 TRGO0
TRIGSEL_INPUT_TIMER2_CH0	trigger input source TIMER2 CH0
TRIGSEL_INPUT_TIMER2_CH1	trigger input source TIMER2 CH1
TRIGSEL_INPUT_TIMER2_CH2	trigger input source TIMER2 CH2
TRIGSEL_INPUT_TIMER2_CH3	trigger input source TIMER2 CH3
TRIGSEL_INPUT_TIMER2_ETI	trigger input source TIMER2 ETI
TRIGSEL_INPUT_TIMER3_TRGO0	trigger input source TIMER3 TRGO0
TRIGSEL_INPUT_TIMER3_CH0	trigger input source TIMER3 CH0
TRIGSEL_INPUT_TIMER3_CH1	trigger input source TIMER3 CH1
TRIGSEL_INPUT_TIMER3_CH2	trigger input source TIMER3 CH2
TRIGSEL_INPUT_TIMER3_CH3	trigger input source TIMER3 CH3
TRIGSEL_INPUT_TIMER3_ETI	trigger input source TIMER3 ETI
TRIGSEL_INPUT_TIMER4_TRGO0	trigger input source TIMER4 TRGO0
TRIGSEL_INPUT_TIMER4_CH0	trigger input source TIMER4 CH0
TRIGSEL_INPUT_TIMER4_CH1	trigger input source TIMER4 CH1
TRIGSEL_INPUT_TIMER4_CH2	trigger input source TIMER4 CH2
TRIGSEL_INPUT_TIMER4_CH3	trigger input source TIMER4 CH3
TRIGSEL_INPUT_TIMER4_ETI	trigger input source TIMER4 ETI
TRIGSEL_INPUT_TIMER5_TRGO0	trigger input source TIMER5 TRGO0
TRIGSEL_INPUT_TIMER6_TRGO0	trigger input source TIMER6 TRGO0
TRIGSEL_INPUT_TIMER7_TRGO0	trigger input source TIMER7 TRGO0
TRIGSEL_INPUT_TIMER7_TRGO1	trigger input source TIMER7 TRGO1

Member name	Function description
TRIGSEL_INPUT_TIMER7_CH0	trigger input source TIMER7 CH0
TRIGSEL_INPUT_TIMER7_CH1	trigger input source TIMER7 CH1
TRIGSEL_INPUT_TIMER7_CH2	trigger input source TIMER7 CH2
TRIGSEL_INPUT_TIMER7_CH3	trigger input source TIMER7 CH3
TRIGSEL_INPUT_TIMER7_MCH0	trigger input source TIMER7 MCH0
TRIGSEL_INPUT_TIMER7_MCH1	trigger input source TIMER7 MCH1
TRIGSEL_INPUT_TIMER7_MCH2	trigger input source TIMER7 MCH2
TRIGSEL_INPUT_TIMER7_MCH3	trigger input source TIMER7 MCH3
TRIGSEL_INPUT_TIMER7_ETI	trigger input source TIMER7 ETI
TRIGSEL_INPUT_TIMER14_TRGO0	trigger input source TIMER14 TRGO0
TRIGSEL_INPUT_TIMER14_CH0	trigger input source TIMER14 CH0
TRIGSEL_INPUT_TIMER14_CH1	trigger input source TIMER14 CH1
TRIGSEL_INPUT_TIMER14_MCH0	trigger input source TIMER14 MCH0
TRIGSEL_INPUT_TIMER15_CH0	trigger input source TIMER15 CH0
TRIGSEL_INPUT_TIMER15_MCH0	trigger input source TIMER15 MCH0
TRIGSEL_INPUT_TIMER16_CH0	trigger input source TIMER16 CH0
TRIGSEL_INPUT_TIMER16_MCH0	trigger input source TIMER16 MCH0
TRIGSEL_INPUT_TIMER19_TRGO0	trigger input source TIMER19 TRGO0
TRIGSEL_INPUT_TIMER19_TRGO1	trigger input source TIMER19 TRGO1
TRIGSEL_INPUT_TIMER19_CH0	trigger input source TIMER19 CH0
TRIGSEL_INPUT_TIMER19_CH1	trigger input source TIMER19 CH1
TRIGSEL_INPUT_TIMER19_CH2	trigger input source TIMER19 CH2
TRIGSEL_INPUT_TIMER19_CH3	trigger input source TIMER19 CH3
TRIGSEL_INPUT_TIMER19_MCH0	trigger input source TIMER19 MCH0
TRIGSEL_INPUT_TIMER19_MCH1	trigger input source TIMER19 MCH1
TRIGSEL_INPUT_TIMER19_MCH2	trigger input source TIMER19 MCH2
TRIGSEL_INPUT_TIMER19_MCH3	trigger input source TIMER19 MCH3
TRIGSEL_INPUT_TIMER19_ETI	trigger input source TIMER19 ETI
TRIGSEL_INPUT_TIMER0_BRKIN0	trigger input source TIMER0 BRKIN0
TRIGSEL_INPUT_TIMER0_BRKIN1	trigger input source TIMER0 BRKIN1
TRIGSEL_INPUT_TIMER0_BRKIN2	trigger input source TIMER0 BRKIN2
TRIGSEL_INPUT_TIMER7_BRKIN0	trigger input source TIMER7 BRKIN0
TRIGSEL_INPUT_TIMER7_BRKIN1	trigger input source TIMER7 BRKIN1
TRIGSEL_INPUT_TIMER7_BRKIN2	trigger input source TIMER7 BRKIN2
TRIGSEL_INPUT_TIMER14_BRKIN0	trigger input source TIMER14 BRKIN0
TRIGSEL_INPUT_TIMER15_BRKIN0	trigger input source TIMER15 BRKIN0
TRIGSEL_INPUT_TIMER16_BRKIN0	trigger input source TIMER16 BRKIN0
TRIGSEL_INPUT_TIMER19_BRKIN0	trigger input source TIMER19 BRKIN0
TRIGSEL_INPUT_TIMER19_BRKIN1	trigger input source TIMER19 BRKIN1
TRIGSEL_INPUT_TIMER19_BRKIN2	trigger input source TIMER19 BRKIN2
TRIGSEL_INPUT_LPTIMER_OUT	trigger input source LPTIMER_OUT

Member name	Function description
TRIGSEL_INPUT_LPTIMER_ETI	trigger input source LPTIMER_ETI
TRIGSEL_INPUT_HRTIMER_SCOUT	trigger input source HRTIMER_SCOUT
TRIGSEL_INPUT_HRTIMER_SCIN	trigger input source HRTIMER_SCIN
TRIGSEL_INPUT_HRTIMER_ADC_TRIG0	trigger input source HRTIMER_ADC_TRIG0
TRIGSEL_INPUT_HRTIMER_ADC_TRIG1	trigger input source HRTIMER_ADC_TRIG1
TRIGSEL_INPUT_HRTIMER_ADC_TRIG2	trigger input source HRTIMER_ADC_TRIG2
TRIGSEL_INPUT_HRTIMER_ADC_TRIG3	trigger input source HRTIMER_ADC_TRIG3
TRIGSEL_INPUT_HRTIMER_ADC_TRIG4	trigger input source HRTIMER_ADC_TRIG4
TRIGSEL_INPUT_HRTIMER_ADC_TRIG5	trigger input source HRTIMER_ADC_TRIG5
TRIGSEL_INPUT_HRTIMER_ADC_TRIG6	trigger input source HRTIMER_ADC_TRIG6
TRIGSEL_INPUT_HRTIMER_ADC_TRIG7	trigger input source HRTIMER_ADC_TRIG7
TRIGSEL_INPUT_HRTIMER_ADC_TRIG8	trigger input source HRTIMER_ADC_TRIG8
TRIGSEL_INPUT_HRTIMER_ADC_TRIG9	trigger input source HRTIMER_ADC_TRIG9
TRIGSEL_INPUT_HRTIMER_DAC_TRIG0	trigger input source HRTIMER_DAC_TRIG0
TRIGSEL_INPUT_HRTIMER_DAC_TRIG1	trigger input source HRTIMER_DAC_TRIG1
TRIGSEL_INPUT_HRTIMER_DAC_TRIG2	trigger input source HRTIMER_DAC_TRIG2
TRIGSEL_INPUT_HRTIMER_ST0_TRIG1	trigger input source HRTIMER_ST0_TRIG1
TRIGSEL_INPUT_HRTIMER_ST1_TRIG1	trigger input source HRTIMER_ST1_TRIG1
TRIGSEL_INPUT_HRTIMER_ST2_TRIG1	trigger input source HRTIMER_ST2_TRIG1
TRIGSEL_INPUT_HRTIMER_ST3_TRIG1	trigger input source HRTIMER_ST3_TRIG1
TRIGSEL_INPUT_HRTIMER_ST4_TRIG1	trigger input source HRTIMER_ST4_TRIG1
TRIGSEL_INPUT_HRTIMER_ST5_TRIG1	trigger input source HRTIMER_ST5_TRIG1
TRIGSEL_INPUT_HRTIMER_ST6_TRIG1	trigger input source HRTIMER_ST6_TRIG1
TRIGSEL_INPUT_HRTIMER_ST7_TRIG1	trigger input source HRTIMER_ST7_TRIG1
TRIGSEL_INPUT_HRTIMER_SYSFLT	trigger input source HRTIMER_SYSFLT
TRIGSEL_INPUT_ADC0_WD0_OUT	trigger input source ADC0 watchdog0 output
TRIGSEL_INPUT_ADC0_WD1_OUT	trigger input source ADC0 watchdog1 output
TRIGSEL_INPUT_ADC0_WD2_OUT	trigger input source ADC0 watchdog2 output
TRIGSEL_INPUT_ADC1_WD0_OUT	trigger input source ADC1 watchdog0 output
TRIGSEL_INPUT_ADC1_WD1_OUT	trigger input source ADC1 watchdog1 output
TRIGSEL_INPUT_ADC1_WD2_OUT	trigger input source ADC1 watchdog2 output
TRIGSEL_INPUT_ADC2_WD0_OUT	trigger input source ADC2 watchdog0 output
TRIGSEL_INPUT_ADC2_WD1_OUT	trigger input source ADC2 watchdog1 output
TRIGSEL_INPUT_ADC2_WD2_OUT	trigger input source ADC2 watchdog2 output
TRIGSEL_INPUT_HXTAL_DIV32_TRIG	trigger input source HXTAL_DIV32
TRIGSEL_INPUT_IRC32K_TRIG	trigger input source IRC32K
TRIGSEL_INPUT_LXTAL_TRIG	trigger input source LXTAL
TRIGSEL_INPUT_CKOUT_TRIG	trigger input source CKOUT
TRIGSEL_INPUT_EXTI2_TRIG	trigger input source EXTI2
TRIGSEL_INPUT_EXTI3_TRIG	trigger input source EXTI3
TRIGSEL_INPUT_EXTI9_TRIG	trigger input source EXTI9

Member name	Function description
TRIGSEL_INPUT_EXTI10_TRIG	trigger input source EXTI10
TRIGSEL_INPUT_EXTI11_TRIG	trigger input source EXTI11
TRIGSEL_INPUT_EXTI15_TRIG	trigger input source EXTI15
TRIGSEL_INPUT_RTC_WAKEUP	trigger input source RTC wakeup
TRIGSEL_INPUT_RTC_ALARM0	trigger input source RTC alarm0
TRIGSEL_INPUT_RTC_ALARM1	trigger input source RTC alarm1
TRIGSEL_INPUT_RTC_TAMP0	trigger input source RTC tamper0
TRIGSEL_INPUT_RTC_TAMP1	trigger input source RTC tamper1
TRIGSEL_INPUT_RTC_TAMP2	trigger input source RTC tamper2
TRIGSEL_INPUT_CMP0_OUT	trigger input source CMP0_OUT
TRIGSEL_INPUT_CMP1_OUT	trigger input source CMP1_OUT
TRIGSEL_INPUT_CMP2_OUT	trigger input source CMP2_OUT
TRIGSEL_INPUT_CMP3_OUT	trigger input source CMP3_OUT
TRIGSEL_INPUT_CMP4_OUT	trigger input source CMP4_OUT
TRIGSEL_INPUT_CMP5_OUT	trigger input source CMP5_OUT
TRIGSEL_INPUT_CMP6_OUT	trigger input source CMP6_OUT
TRIGSEL_INPUT_CMP7_OUT	trigger input source CMP7_OUT
TRIGSEL_INPUT_CLA_OUT0	trigger input source CLA_OUT0
TRIGSEL_INPUT_CLA_OUT1	trigger input source CLA_OUT1
TRIGSEL_INPUT_CLA_OUT2	trigger input source CLA_OUT2
TRIGSEL_INPUT_CLA_OUT3	trigger input source CLA_OUT3
TRIGSEL_INPUT_HRTIMER_ST0_TRIG0	trigger input source HRTIMER_ST0_TRIG0
TRIGSEL_INPUT_HRTIMER_ST1_TRIG0	trigger input source HRTIMER_ST1_TRIG0
TRIGSEL_INPUT_HRTIMER_ST2_TRIG0	trigger input source HRTIMER_ST2_TRIG0
TRIGSEL_INPUT_HRTIMER_ST3_TRIG0	trigger input source HRTIMER_ST3_TRIG0
TRIGSEL_INPUT_HRTIMER_ST4_TRIG0	trigger input source HRTIMER_ST4_TRIG0
TRIGSEL_INPUT_HRTIMER_ST5_TRIG0	trigger input source HRTIMER_ST5_TRIG0
TRIGSEL_INPUT_HRTIMER_ST6_TRIG0	trigger input source HRTIMER_ST6_TRIG0
TRIGSEL_INPUT_HRTIMER_ST7_TRIG0	trigger input source HRTIMER_ST7_TRIG0
TRIGSEL_INPUT_HRTIMER_ST0_CH0	trigger input source HRTIMER slave0 CH0
TRIGSEL_INPUT_HRTIMER_ST0_CH1	trigger input source HRTIMER slave0 CH1
TRIGSEL_INPUT_HRTIMER_ST1_CH0	trigger input source HRTIMER slave1 CH0
TRIGSEL_INPUT_HRTIMER_ST1_CH1	trigger input source HRTIMER slave1 CH1
TRIGSEL_INPUT_HRTIMER_ST2_CH0	trigger input source HRTIMER slave2 CH0
TRIGSEL_INPUT_HRTIMER_ST2_CH1	trigger input source HRTIMER slave2 CH1
TRIGSEL_INPUT_HRTIMER_ST3_CH0	trigger input source HRTIMER slave3 CH0
TRIGSEL_INPUT_HRTIMER_ST3_CH1	trigger input source HRTIMER slave3 CH1
TRIGSEL_INPUT_HRTIMER_ST4_CH0	trigger input source HRTIMER slave4 CH0
TRIGSEL_INPUT_HRTIMER_ST4_CH1	trigger input source HRTIMER slave4 CH1
TRIGSEL_INPUT_HRTIMER_ST5_CH0	trigger input source HRTIMER slave5 CH0
TRIGSEL_INPUT_HRTIMER_ST5_CH1	trigger input source HRTIMER slave5 CH1

Member name	Function description
TRIGSEL_INPUT_HRTIMER_ST6_CH0	trigger input source HRTIMER slave6 CH0
TRIGSEL_INPUT_HRTIMER_ST6_CH1	trigger input source HRTIMER slave6 CH1
TRIGSEL_INPUT_HRTIMER_ST7_CH0	trigger input source HRTIMER slave7 CH0
TRIGSEL_INPUT_HRTIMER_ST7_CH1	trigger input source HRTIMER slave7 CH1
TRIGSEL_INPUT_ADC0_CONV	trigger input source ADC0 CONV
TRIGSEL_INPUT_ADC1_CONV	trigger input source ADC1 CONV
TRIGSEL_INPUT_ADC2_CONV	trigger input source ADC2 CONV
TRIGSEL_INPUT_ADC3_CONV	trigger input source ADC3 CONV
TRIGSEL_INPUT_LCKM_OUT	trigger input source LCKM_OUT

### Enum trigsel\_periph\_enum

**Table 3-1286. 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_ROUTRG	output target peripheral ADC0_ROUTRG
TRIGSEL_OUTPUT_ADC0_INSTRG	output target peripheral ADC0_INSTRG
TRIGSEL_OUTPUT_ADC1_ROUTRG	output target peripheral ADC1_ROUTRG
TRIGSEL_OUTPUT_ADC1_INSTRG	output target peripheral ADC1_INSTRG
TRIGSEL_OUTPUT_ADC2_ROUTRG	output target peripheral ADC2_ROUTRG
TRIGSEL_OUTPUT_ADC2_INSTRG	output target peripheral ADC2_INSTRG
TRIGSEL_OUTPUT_ADC3_ROUTRG	output target peripheral ADC3_ROUTRG
TRIGSEL_OUTPUT_ADC3_INSTRG	output target peripheral ADC3_INSTRG
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_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_TIMER14_BRKIN0	output target peripheral TIMER14_BRKIN0
TRIGSEL_OUTPUT_TIMER15_BRKIN0	output target peripheral TIMER15_BRKIN0
TRIGSEL_OUTPUT_TIMER16_BRKIN0	output target peripheral TIMER16_BRKIN0
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



Member name	Function description
TRIGSEL_OUTPUT_CAN0_EX_TIME_TICK	output target peripheral CAN0_EX_TIME_TICK
TRIGSEL_OUTPUT_CAN1_EX_TIME_TICK	output target peripheral CAN1_EX_TIME_TICK
TRIGSEL_OUTPUT_CAN2_EX_TIME_TICK	output target peripheral CAN2_EX_TIME_TICK
TRIGSEL_OUTPUT_TIMER0_ETI	output target peripheral TIMER0_ETI
TRIGSEL_OUTPUT_TIMER1_ETI	output target peripheral TIMER1_ETI
TRIGSEL_OUTPUT_TIMER2_ETI	output target peripheral TIMER2_ETI
TRIGSEL_OUTPUT_TIMER3_ETI	output target peripheral TIMER3_ETI
TRIGSEL_OUTPUT_TIMER4_ETI	output target peripheral TIMER4_ETI
TRIGSEL_OUTPUT_TIMER7_ETI	output target peripheral TIMER7_ETI
TRIGSEL_OUTPUT_TIMER19_ETI	output target peripheral TIMER19_ETI
TRIGSEL_OUTPUT_HPDI_ITRG	output target peripheral HPDI_ITRG
TRIGSEL_OUTPUT_TIMER0_ITI14	output target peripheral TIMER0_ITI14
TRIGSEL_OUTPUT_TIMER1_ITI14	output target peripheral TIMER1_ITI14
TRIGSEL_OUTPUT_TIMER2_ITI14	output target peripheral TIMER2_ITI14
TRIGSEL_OUTPUT_TIMER3_ITI14	output target peripheral TIMER3_ITI14
TRIGSEL_OUTPUT_TIMER4_ITI14	output target peripheral TIMER4_ITI14
TRIGSEL_OUTPUT_TIMER7_ITI14	output target peripheral TIMER7_ITI14
TRIGSEL_OUTPUT_TIMER14_ITI14	output target peripheral TIMER14_ITI14
TRIGSEL_OUTPUT_TIMER19_ITI14	output target peripheral TIMER19_ITI14
TRIGSEL_OUTPUT_DAC0_EXTRIG0	output target peripheral DAC0_EXTRIG0
TRIGSEL_OUTPUT_DAC0_EXTRIG1	output target peripheral DAC0_EXTRIG1
TRIGSEL_OUTPUT_DAC1_EXTRIG0	output target peripheral DAC1_EXTRIG0
TRIGSEL_OUTPUT_DAC1_EXTRIG1	output target peripheral DAC1_EXTRIG1
TRIGSEL_OUTPUT_DAC2_EXTRIG0	output target peripheral DAC2_EXTRIG0
TRIGSEL_OUTPUT_DAC2_EXTRIG1	output target peripheral DAC2_EXTRIG1
TRIGSEL_OUTPUT_DAC3_EXTRIG0	output target peripheral DAC3_EXTRIG0
TRIGSEL_OUTPUT_DAC3_EXTRIG1	output target peripheral DAC3_EXTRIG1
TRIGSEL_OUTPUT_DAC0_ST_EXTRIG0	output target peripheral DAC0_ST_EXTRIG0
TRIGSEL_OUTPUT_DAC0_ST_EXTRIG1	output target peripheral DAC0_ST_EXTRIG1
TRIGSEL_OUTPUT_DAC1_ST_EXTRIG0	output target peripheral DAC1_ST_EXTRIG0
TRIGSEL_OUTPUT_DAC1_ST_EXTRIG1	output target peripheral DAC1_ST_EXTRIG1
TRIGSEL_OUTPUT_DAC2_ST_EXTRIG0	output target peripheral DAC2_ST_EXTRIG0
TRIGSEL_OUTPUT_DAC2_ST_EXTRIG1	output target peripheral DAC2_ST_EXTRIG1
TRIGSEL_OUTPUT_DAC3_ST_EXTRIG0	output target peripheral DAC3_ST_EXTRIG0
TRIGSEL_OUTPUT_DAC3_ST_EXTRIG1	output target peripheral DAC3_ST_EXTRIG1
TRIGSEL_OUTPUT_CLA_IN0	output target peripheral CLA_IN0
TRIGSEL_OUTPUT_CLA_IN1	output target peripheral CLA_IN1
TRIGSEL_OUTPUT_CLA_IN2	output target peripheral CLA_IN2



Member name	Function description
TRIGSEL_OUTPUT_CLA_IN3	output target peripheral CLA_IN3
TRIGSEL_OUTPUT_CLA_IN4	output target peripheral CLA_IN4
TRIGSEL_OUTPUT_CLA_IN5	output target peripheral CLA_IN5
TRIGSEL_OUTPUT_CLA_IN6	output target peripheral CLA_IN6
TRIGSEL_OUTPUT_CLA_IN7	output target peripheral CLA_IN7
TRIGSEL_OUTPUT_CLA_IN8	output target peripheral CLA_IN8
TRIGSEL_OUTPUT_CLA_IN9	output target peripheral CLA_IN9
TRIGSEL_OUTPUT_CLA_IN10	output target peripheral CLA_IN10
TRIGSEL_OUTPUT_CLA_IN11	output target peripheral CLA_IN11

### trigsel\_deinit

The description of trigsel\_deinit is shown as below:

**Table 3-1287. Function trigsel\_deinit**

<b>Function name</b>	trigsel_deinit
<b>Function prototype</b>	void trigsel_deinit(void);
<b>Function descriptions</b>	Deinitialize TRIGSEL
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize TRIGSEL */
```

```
trigsel_deinit();
```

### trigsel\_init

The description of trigsel\_init is shown as below:

**Table 3-1288. Function trigsel\_init**

<b>Function name</b>	trigsel_init
<b>Function prototype</b>	void trigsel_init(trigsel_periph_enum target_periph, trigsel_source_enum trigger_source);
<b>Function descriptions</b>	Set the trigger input signal for target peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>target_periph</b>	Target peripheral value, refer to <a href="#">Table 3-1286. Enum trigsel_periph_enum</a>
<b>Input parameter{in}</b>	
<b>trigger_source</b>	Trigger source value, refer to <a href="#">Table 3-1285. Enum trigsel_source_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0_CH2 to trigger ADC0 */
trigsels_init(TRIGSEL_OUTPUT_ADC0_ROUTRG, TRIGSEL_INPUT_TIMER0_CH2);
```

### trigsels\_trigger\_source\_get

The description of trigsels\_trigger\_source\_get is shown as below:

**Table 3-1289. Function trigsels\_trigger\_source\_get**

<b>Function name</b>	trigsels_trigger_source_get
<b>Function prototype</b>	trigsels_source_enum trigsels_trigger_source_get(trigsels_periph_enum target_periph);
<b>Function descriptions</b>	Get the trigger input signal for target peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_periph</b>	Target peripheral value, refer to <a href="#">Table 3-1286. Enum trigsels_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>trigger_source</b>	Trigger source value, refer to <a href="#">Table 3-1285. Enum trigsels_source_enum</a>

Example:

```
/* get the trigger input signal for ADC0 */
trigsels_source_enum input_signal;
input_signal = trigsels_trigger_source_get(TRIGSEL_OUTPUT_ADC0_ROUTRG);
```

### trigsels\_register\_lock\_set

The description of trigsels\_register\_lock\_set is shown as below:

**Table 3-1290. Function trigsels\_register\_lock\_set**

<b>Function name</b>	trigsels_register_lock_set
----------------------	----------------------------

<b>Function prototype</b>	void trigsels_register_lock_set(trigsels_periph_enum target_periph);
<b>Function descriptions</b>	Lock the trigger register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_periph</b>	Target peripheral value, refer to <a href="#">Table 3-1286. Enum trigsels_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* lock the trigger register for ADC0 */
trigsels_register_lock_set(TRIGSELS_OUTPUT_ADC0_ROUTRG);
```

### trigsels\_register\_lock\_get

The description of trigsels\_register\_lock\_get is shown as below:

**Table 3-1291. Function trigsels\_register\_lock\_get**

<b>Function name</b>	trigsels_register_lock_get
<b>Function prototype</b>	FlagStatus trigsels_register_lock_get(trigsels_periph_enum target_periph);
<b>Function descriptions</b>	Get the trigger register lock status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_periph</b>	Target peripheral value, refer to <a href="#">Table 3-1286. Enum trigsels_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the trigger register lock status of ADC0 */
FlagStatus status;
status = trigsels_register_lock_get(TRIGSELS_OUTPUT_ADC0_ROUTRG);
```

## 3.33. TRNG

The true random number generator (TRNG) module can generate a 32-bit value using

continuous analog noise and it has been pre-certified NIST SP800-90B. The TRNG registers are listed in chapter [3.33.1](#). the TRNG firmware functions are introduced in chapter [3.33.2](#).

### 3.33.1. Descriptions of Peripheral registers

TRNG registers are listed in the table shown as below:

**Table 3-1292. TRNG Registers**

Registers	Descriptions
TRNG_CTL	control register
TRNG_STAT	status register
TRNG_DATA	data register
TRNG_HTCFG	health tests configure register

### 3.33.2. Descriptions of Peripheral functions

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

**Table 3-1293. TRNG firmware function**

Function name	Function description
trng_deinit	deinitialize the TRNG
trng_enable	enable the TRNG interface
trng_disable	disable the TRNG interface
trng_lock	lock the TRNG control bits
trng_mode_config	Configure TRNG working mode
trng_nist_seed_config	Configure TRNG NIST seed mode
trng_postprocessing_enable	enable the TRNG post-processing module
trng_postprocessing_disable	disable the TRNG post-processing module
trng_conditioning_enable	enable the TRNG conditioning module
trng_conditioning_disable	disable the TRNG conditioning module
trng_conditioning_input_bitwidth	configure TRNG conditioning module input bitwidth
trng_conditioning_output_bitwidth	configure TRNG conditioning module output bitwidth
trng_replace_test_enable	enable TRNG replace test
trng_replace_test_disable	disable TRNG replace test
trng_hash_init_enable	enable hash algorithm init when conditioning module enabled
trng_hash_init_disable	disable hash algorithm init when conditioning module enabled
trng_powermode_config	configure TRNG analog power mode
trng_clockdiv_config	configure TRNG clock divider
trng_clockerror_detection_enable	enable the TRNG clock error detection
trng_clockerror_detection_disable	disable the TRNG clock error detection
trng_get_true_random_data	get the true random data
trng_conditioning_reset_enable	enable the conditioning logic reset
trng_conditioning_reset_disable	disable the conditioning logic reset
trng_conditioning_algo_config	configure the conditioning module hash algorithm

Function name	Function description
trng_health_tests_config	configure health tests default value
trng_flag_get	get the TRNG status flags
trng_interrupt_enable	the TRNG interrupt enable
trng_interrupt_disable	the TRNG interrupt disable
trng_interrupt_flag_get	get the TRNG interrupt flags
trng_interrupt_flag_clear	clear the TRNG interrupt flags

### Enum trng\_inmod\_enum

Table 3-1294. Enum trng\_inmod\_enum

Member name	Function description
TRNG_INMOD_256BIT	conditioning module input bitwidth 256bits
TRNG_INMOD_440BIT	conditioning module input bitwidth 440bits

### Enum trng\_outmod\_enum

Table 3-1295. Enum trng\_outmod\_enum

Member name	Function description
TRNG_OUTMOD_128BIT	conditioning module output bitwidth 128bits
TRNG_OUTMOD_256BIT	conditioning module output bitwidth 256bits

### Enum trng\_modsel\_enum

Table 3-1296. Enum trng\_modsel\_enum

Member name	Function description
TRNG_MODSEL_LFSR	TRNG working in LFSR mode
TRNG_MODSEL_NIST	TRNG working in NIST mode

### Enum trng\_nist\_seed\_enum

Table 3-1297. Enum trng\_nist\_seed\_enum

Member name	Function description
TRNG_MODSEL_LFSR	TRNG working in LFSR mode
TRNG_MODSEL_NIST	TRNG working in NIST mode

### Enum trng\_flag\_enum

Table 3-1298. Enum trng\_flag\_enum

Member name	Function description
TRNG_FLAG_DRDY	random data ready status
TRNG_FLAG_CECS	clock error current status
TRNG_FLAG_SECS	seed error current status

## Enum trng\_int\_flag\_enum

**Table 3-1299. Enum trng\_int\_flag\_enum**

Member name	Function description
TRNG_INT_FLAG_CEIF	clock error interrupt flag
TRNG_INT_FLAG_SEIF	seed error interrupt flag

## trng\_deinit

The description of trng\_deinit is shown as below:

**Table 3-1300. Function trng\_deinit**

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

Example:

```
/* TRNG deinit */
```

```
trng_deinit( );
```

## trng\_enable

The description of trng\_enable is shown as below:

**Table 3-1301. Function trng\_enable**

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

Example:

```
/* enable the TRNG interface */
```

```
trng_enable( );
```

### trng\_disable

The description of trng\_disable is shown as below:

**Table 3-1302. Function trng\_disable**

<b>Function name</b>	trng_disable
<b>Function prototype</b>	void trng_disable(void);
<b>Function descriptions</b>	disable the TRNG interface
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the TRNG interface */
```

```
trng_disable( );
```

### trng\_lock

The description of trng\_lock is shown as below:

**Table 3-1303. Function trng\_lock**

<b>Function name</b>	trng_lock
<b>Function prototype</b>	void trng_lock (void);
<b>Function descriptions</b>	lock the TRNG control bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* lock the TRNG control bits */
```

```
trng_lock( );
```

### trng\_mode\_config

The description of trng\_mode\_config is shown as below:

**Table 3-1304. Function trng\_mode\_config**

<b>Function name</b>	trng_mode_config
<b>Function prototype</b>	void trng_mode_config(trng_modsel_enum mode_select);
<b>Function descriptions</b>	configure TRNG working mode
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mode_select</b>	-
TRNG_MODSEL_LFSR	TRNG working in LFSR mode
TRNG_MODSEL_NIST	TRNG working in NIST mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_nist_seed_config(TRNG_NIST_SEED_ANALOG);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();
```

### trng\_nist\_seed\_config

The description of trng\_nist\_seed\_config is shown as below:



Table 3-1305. Function `trng_nist_seed_config`

<b>Function name</b>	<code>trng_nist_seed_config</code>
<b>Function prototype</b>	<code>void trng_nist_seed_config(trng_nist_seed_enum seed_select);</code>
<b>Function descriptions</b>	configure TRNG NIST seed mode
<b>Precondition</b>	<code>trng_mode_config(TRNG_MODSEL_NIST)</code>
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>seed_select</b>	-
<code>TRNG_NIST_SEED_ANALOG</code>	Select annlog as seed in NIST mode
<code>TRNG_NIST_SEED_LFSR</code>	Select LFSR as seed in NIST mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_nist_seed_config(TRNG_NIST_SEED_ANALOG);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### **trng\_postprocessing\_enable**

The description of `trng_postprocessing_enable` is shown as below:

Table 3-1306. Function `trng_postprocessing_enable`

<b>Function name</b>	<code>trng_postprocessing_enable</code>
<b>Function prototype</b>	<code>void trng_postprocessing_enable(void);</code>

<b>Function descriptions</b>	enable the TRNG post-processing module
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_postprocessing\_disable

The description of trng\_postprocessing\_disable is shown as below:

**Table 3-1307. Function trng\_postprocessing\_disable**

<b>Function name</b>	trng_postprocessing_disable
<b>Function prototype</b>	void trng_postprocessing_disable(void);
<b>Function descriptions</b>	disable the TRNG post-processing module
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_disable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_conditioning\_enable

The description of trng\_conditioning\_enable is shown as below:

**Table 3-1308. Function trng\_conditioning\_enable**

<b>Function name</b>	trng_conditioning_enable
<b>Function prototype</b>	void trng_conditioning_enable(void);
<b>Function descriptions</b>	enable the TRNG conditioning module
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

```

```

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_conditioning\_disable

The description of trng\_conditioning\_disable is shown as below:

**Table 3-1309. Function trng\_conditioning\_disable**

<b>Function name</b>	trng_conditioning_disable
<b>Function prototype</b>	void trng_conditioning_disable(void);
<b>Function descriptions</b>	disable the TRNG conditioning module
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_disable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

## trng\_conditioning\_input\_bitwidth

The description of trng\_conditioning\_input\_bitwidth is shown as below:

**Table 3-1310. Function trng\_disable**

<b>Function name</b>	trng_conditioning_input_bitwidth
<b>Function prototype</b>	void trng_conditioning_input_bitwidth(trng_inmod_enum input_bitwidth);
<b>Function descriptions</b>	configure TRNG conditioning module input bitwidth
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>input_bitwidth</b>	refer to enum <a href="#">Table 3-1294. Enum trng_inmod_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

## trng\_conditioning\_output\_bitwidth

The description of trng\_conditioning\_output\_bitwidth is shown as below:

**Table 3-1311. Function trng\_conditioning\_output\_bitwidth**

<b>Function name</b>	trng_conditioning_output_bitwidth
<b>Function prototype</b>	void trng_conditioning_output_bitwidth(trng_outmod_enum output_bitwidth);
<b>Function descriptions</b>	configure TRNG conditioning module output bitwidth
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-

Input parameter{in}	
output_bitwidth	refer to enum <a href="#">Table 3-1295. Enum trng_outmod_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_replace\_test\_enable

The description of trng\_replace\_test\_enable is shown as below:

**Table 3-1312. Function trng\_replace\_test\_enable**

Function name	trng_replace_test_enable
Function prototype	void trng_replace_test_enable(void);
Function descriptions	enable TRNG replace test
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable TRNG replace test */

```

```

trng_deinit();

trng_conditioning_reset_enable();

trng_replace_test_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_replace\_test\_disable

The description of trng\_replace\_test\_disable is shown as below:

**Table 3-1313. Function trng\_replace\_test\_disable**

<b>Function name</b>	trng_replace_test_disable
<b>Function prototype</b>	void trng_replace_test_disable(void);
<b>Function descriptions</b>	disable TRNG replace test
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* disable TRNG replace test */

trng_deinit();

trng_conditioning_reset_enable();

trng_replace_test_disable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_hash\_init\_enable

The description of trng\_hash\_init\_enable is shown as below:

**Table 3-1314. Function trng\_hash\_init\_enable**

<b>Function name</b>	trng_hash_init_enable
<b>Function prototype</b>	void trng_hash_init_enable(void);
<b>Function descriptions</b>	enable hash algorithm init when conditioning module enabled
<b>Precondition</b>	trng_conditioning_reset_enable / trng_conditioning_enable

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

Example:

```
/* enable hash algorithm init when conditioning module enabled */
```

```
trng_deinit();
```

```
trng_conditioning_reset_enable();
```

```
trng_conditioning_enable();
```

```
trng_hash_init_enable();
```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_hash\_init\_disable

The description of trng\_hash\_init\_disable is shown as below:

**Table 3-1315. Function trng\_hash\_init\_disable**

Function name	trng_hash_init_disable
Function prototype	void trng_hash_init_disable(void);
Function descriptions	disable hash algorithm init when conditioning module enabled
Precondition	trng_conditioning_reset_enable / trng_conditioning_enable
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TRNG interface */
```

```
trng_deinit();
```

```
trng_conditioning_reset_enable();
```

```
trng_conditioning_enable();
```



```
trng_hash_init_disable();

trng_enable();

trng_conditioning_reset_disable();
```

### trng\_powermode\_config

The description of trng\_powermode\_config is shown as below:

**Table 3-1316. Function trng\_powermode\_config**

<b>Function name</b>	trng_powermode_config
<b>Function prototype</b>	void trng_powermode_config(uint32_t powermode);
<b>Function descriptions</b>	configure TRNG analog power mode
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>powermode</b>	the power mode selection
<i>TRNG_NR_ULATRL0</i> <i>W</i>	TRNG analog power mode ulatrlow
<i>TRNG_NR_LOW</i>	TRNG analog power mode low
<i>TRNG_NR_MEDIUM</i>	TRNG analog power mode medium
<i>TRNG_NR_HIGH</i>	TRNG analog power mode high
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TRNG analog power mode as high */

trng_deinit();

trng_conditioning_reset_enable();

trng_powermode_config(TRNG_NR_HIGH);

trng_enable();

trng_conditioning_reset_disable();
```

### trng\_clockdiv\_config

The description of trng\_clockdiv\_config is shown as below:

**Table 3-1317. Function trng\_disable**

<b>Function name</b>	trng_clockdiv_config
<b>Function prototype</b>	void trng_clockdiv_config(uint32_t clkdiv);

<b>Function descriptions</b>	configure TRNG clock divider
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clkdiv</b>	the TRNG clock divider
<i>TRNG_CLK_DIVx</i>	clock division coefficient x, x = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TRNG clock frequency division coefficient to 64 */
```

```
trng_deinit();
```

```
trng_conditioning_reset_enable();
```

```
trng_clockdiv_config(TRNG_CLK_DIV64);
```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_clockerror\_detection\_enable

The description of trng\_clockerror\_detection\_enable is shown as below:

**Table 3-1318. Function trng\_clockerror\_detection\_enable**

<b>Function name</b>	trng_clockerror_detection_enable
<b>Function prototype</b>	void trng_clockerror_detection_enable(void);
<b>Function descriptions</b>	enable the TRNG clock error detection
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the TRNG clock error detection */
```

```
trng_deinit();
```

```
trng_conditioning_reset_enable();

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();
```

### trng\_clockerror\_detection\_disable

The description of trng\_clockerror\_detection\_disable is shown as below:

**Table 3-1319. Function trng\_clockerror\_detection\_disable**

<b>Function name</b>	trng_clockerror_detection_disable
<b>Function prototype</b>	void trng_clockerror_detection_disable(void);
<b>Function descriptions</b>	disable the TRNG clock error detection
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the TRNG clock error detection */

trng_deinit();

trng_conditioning_reset_enable();

trng_clockerror_detection_disable();

trng_enable();

trng_conditioning_reset_disable();
```

### trng\_get\_true\_random\_data

The description of trng\_get\_true\_random\_data is shown as below:

**Table 3-1320. Function trng\_get\_true\_random\_data**

<b>Function name</b>	trng_get_true_random_data
<b>Function prototype</b>	uint32_t trng_get_true_random_data(void);
<b>Function descriptions</b>	get the true random data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

-	-
Output parameter{out}	
-	-
Return value	
uint32_t	0x0 – 0xFFFFFFFF

Example:

```
/* get the true random data */

uint32_t data = 0;

data = trng_get_true_random_data();
```

### trng\_conditioning\_reset\_enable

The description of trng\_conditioning\_reset\_enable is shown as below:

**Table 3-1321. Function trng\_conditioning\_reset\_enable**

<b>Function name</b>	trng_conditioning_reset_enable
<b>Function prototype</b>	void trng_conditioning_reset_enable (void)
<b>Function descriptions</b>	enable the conditioning logic reset
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();
```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_conditioning\_reset\_disable

The description of trng\_conditioning\_reset\_disable is shown as below:

**Table 3-1322. Function trng\_conditioning\_reset\_disable**

<b>Function name</b>	trng_conditioning_reset_disable
<b>Function prototype</b>	void trng_conditioning_reset_disable(void)
<b>Function descriptions</b>	disable the conditioning logic reset
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();
```

### trng\_conditioning\_algo\_config

The description of trng\_conditioning\_algo\_config is shown as below:

**Table 3-1323. Function trng\_conditioning\_algo\_config**

<b>Function name</b>	trng_conditioning_algo_config
----------------------	-------------------------------

<b>Function prototype</b>	void trng_conditioning_algo_config(uint32_t module_algo)
<b>Function descriptions</b>	configure the conditioning module hash algorithm
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>module_algo</b>	module hash algorithm
TRNG_ALGO_SHA1	TRNG conditioning module hash SHA1
TRNG_ALGO_MD5	TRNG conditioning module hash MD5
TRNG_ALGO_SHA224	TRNG conditioning module hash SHA224
TRNG_ALGO_SHA256	TRNG conditioning module hash SHA256
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_conditioning_algo_config(TRNG_ALGO_SHA1);

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_health\_tests\_config

The description of trng\_health\_tests\_config is shown as below:

**Table 3-1324. Function trng\_health\_tests\_config**

<b>Function name</b>	trng_health_tests_config
<b>Function prototype</b>	void trng_health_tests_config(uint32_t adpo_threshold, uint8_t rep_threshold)
<b>Function descriptions</b>	configure health tests default value

<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adpo_threshold</b>	adaptive proportion test threshold value
<b>Input parameter{in}</b>	
<b>rep_threshold</b>	repetitive (00/11) test threshold value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_health_tests_config(700, 50);

trng_enable();

```

### trng\_flag\_get

The description of trng\_flag\_get is shown as below:

**Table 3-1325. Function trng\_flag\_get**

<b>Function name</b>	trng_flag_get
<b>Function prototype</b>	FlagStatus trng_flag_get(trng_flag_enum flag);
<b>Function descriptions</b>	get the trng status flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	trng status flag, refer to <a href="#">Table 3-1298. Enum trng_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

<b>FlagStatus</b>	SET or RESET
-------------------	--------------

Example:

```
/* get the trng status flags */
```

```
FlagStatus status;
```

```
status = trng_flag_get(TRNG_FLAG_DRDY);
```

### trng\_interrupt\_enable

The description of trng\_interrupt\_enable is shown as below:

**Table 3-1326. Function trng\_interrupt\_enable**

<b>Function name</b>	trng_interrupt_enable
<b>Function prototype</b>	void trng_interrupt_enable(void);
<b>Function descriptions</b>	enable the TRNG interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the TRNG interrupt */
```

```
trng_interrupt_enable( );
```

### trng\_interrupt\_disable

The description of trng\_interrupt\_disable is shown as below:

**Table 3-1327. Function trng\_interrupt\_disable**

<b>Function name</b>	trng_interrupt_disable
<b>Function prototype</b>	void trng_interrupt_disable(void);
<b>Function descriptions</b>	disable the TRNG interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	



-	-
---	---

Example:

```
/* disable the TRNG interrupt */
```

```
trng_interrupt_disable( );
```

### trng\_interrupt\_flag\_get

The description of trng\_interrupt\_flag\_get is shown as below:

**Table 3-1328. Function trng\_interrupt\_flag\_get**

<b>Function name</b>	trng_interrupt_flag_get
<b>Function prototype</b>	FlagStatus trng_interrupt_flag_get(trng_int_flag_enum int_flag)
<b>Function descriptions</b>	get the trng interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	trng interrupt flag, refer to <a href="#">Table 3-1299. Enum trng_int_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the trng interrupt flag */
```

```
FlagStatus status = RESET;
```

```
status = trng_interrupt_flag_get(TRNG_INT_FLAG_CEIF);
```

### trng\_interrupt\_flag\_clear

The description of trng\_interrupt\_flag\_clear is shown as below:

**Table 3-1329. Function trng\_interrupt\_flag\_clear**

<b>Function name</b>	trng_interrupt_flag_clear
<b>Function prototype</b>	void trng_interrupt_flag_clear(trng_int_flag_enum int_flag)
<b>Function descriptions</b>	clear the trng interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	trng interrupt flag, refer to <a href="#">Table 3-1299. Enum trng_int_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/*clear the trng interrupt flag */
```

```
trng_interrupt_flag_clear(TRNG_INT_FLAG_CEIF);
```

## 3.34. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter [3.34.1](#), the USART firmware functions are introduced in chapter [3.34.2](#).

### 3.34.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

**Table 3-1330. USART Registers**

Registers	Descriptions
USART_CTL0	control register 0
USART_CTL1	control register 1
USART_CTL2	control register 2
USART_BAUD	baud rate generator register
USART_GP	prescaler and guard time configuration register
USART_RT	receiver timeout register
USART_CMD	command register
USART_STAT	status register
USART_INTC	interrupt status clear register
USART_RDATA	receive data register
USART_TDATA	transmit data register
USART_CHC	coherence control register
USART_FCS	FIFO control and status register

### 3.34.2. Descriptions of Peripheral functions

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

**Table 3-1331. 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

Function name	Function description
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_0_match_mode_enable	enable address 0 match mode
usart_address_0_match_mode_disable	disable address 0 match mode
usart_address_1_match_mode_enable	enable address 1 match mode
usart_address_1_match_mode_disable	disable address 1 match mode
usart_address_0_config	configure address 0 of the USART
usart_address_1_config	configure address 1 of the USART
usart_address_0_detection_mode_config	configure address 0 detection mode
usart_address_1_detection_mode_config	configure address 1 detection mode
usart_mute_mode_enable	enable mute mode
usart_mute_mode_disable	disable mute mode
usart_mute_mode_wakeup_config	configure wakeup method in mute mode
usart_lin_mode_enable	enable LIN mode
usart_lin_mode_disable	disable LIN mode
usart_lin_break_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

Function name	Function description
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 for reception
usart_dma_transmit_config	configure USART DMA for transmission
usart_reception_error_dma_disable	disable DMA on reception error
usart_reception_error_dma_enable	enable DMA on reception error
usart_wakeup_enable	enable USART to wakeup the mcu from deep-sleep mode
usart_wakeup_disable	disable USART to wakeup the mcu from deep-sleep mode
usart_wakeup_mode_config	configure the USART wakeup mode from deep-sleep mode
usart_fifo_enable	enable FIFO
usart_fifo_disable	disable FIFO
usart_transmit_fifo_threshold_config	configure transmit FIFO threshold
usart_receive_fifo_threshold_config	configure receive FIFO threshold
usart_receive_fifo_counter_number	read receive FIFO counter number
usart_flag_get	get USART status
usart_flag_clear	clear USART status

Function name	Function description
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 3-1332. 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_AM0	ADDR0 match flag
USART_FLAG_BSY	busy flag
USART_FLAG_AM1	ADDR1 match 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_TFNF	transmit FIFO not full
USART_FLAG_TC	transmission complete
USART_FLAG_RBNE	read data buffer not empty
USART_FLAG_RFNE	receive FIFO 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_TFTIF	transmit FIFO threshold interrupt flag
USART_FLAG_TFEIF	transmit FIFO empty interrupt flag
USART_FLAG_RFTIF	receive FIFO threshold interrupt flag
USART_FLAG_RFFIF	receive FIFO full interrupt flag
USART_FLAG_RFF	receive FIFO full flag
USART_FLAG_RFE	receive FIFO empty flag
USART_FLAG_TFF	transmit FIFO full flag
USART_FLAG_TFE	transmit FIFO empty flag
USART_FLAG_TFT	transmit FIFO threshold reach flag

Member name	Function description
USART_FLAG_RFT	receive FIFO threshold reach flag

### Enum usart\_interrupt\_flag\_enum

**Table 3-1333. Enum usart\_interrupt\_flag\_enum**

Member name	Function description
USART_INT_FLAG_AM1	address 1 match interrupt and flag
USART_INT_FLAG_EB	end of block interrupt and flag
USART_INT_FLAG_RT	receiver timeout interrupt and flag
USART_INT_FLAG_AM0	address 0 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_TFNF	transmit FIFO not full 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_RFNE	receive FIFO 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_TFT	transmit FIFO threshold reach interrupt and flag
USART_INT_FLAG_TFE	transmit FIFO empty interrupt and flag
USART_INT_FLAG_RFT	receive FIFO threshold reach interrupt and flag
USART_INT_FLAG_RFF	receive FIFO full interrupt and flag

### Enum usart\_interrupt\_enum

**Table 3-1334. Enum usart\_interrupt\_enum**

Member name	Function description
USART_INT_AM1	address 1 match interrupt
USART_INT_EB	end of block interrupt
USART_INT_RT	receiver timeout interrupt
USART_INT_AM0	Address 0 match interrupt
USART_INT_PERR	parity error interrupt
USART_INT_TBE	transmitter buffer empty interrupt
USART_INT_TFNF	transmit FIFO not full interrupt

Member name	Function description
USART_INT_TC	transmission complete interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt
USART_INT_RFNE	receive FIFO 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_TFE	transmit FIFO empty interrupt
USART_INT_TFT	transmit FIFO threshold interrupt
USART_INT_RFT	receive FIFO threshold interrupt
USART_INT_RFF	receive FIFO full interrupt

### Enum usart\_invert\_enum

Table 3-1335. 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 3-1336. 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
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* reset USART0 */
usart_deinit(USART0);
```

### usart\_baudrate\_set

The description of usart\_baudrate\_set is shown as below:

**Table 3-1337. Function usart\_baudrate\_set**

<b>Function name</b>	usart_baudrate_set
<b>Function prototype</b>	void usart_baudrate_set(uint32_t usart_periph, uint32_t baudval);
<b>Function descriptions</b>	configure USART baud rate value
<b>Precondition</b>	-
<b>The called functions</b>	rcu_clock_freq_get
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2,
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>baudval</b>	baud rate value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1338. Function usart\_parity\_config**

<b>Function name</b>	usart_parity_config
<b>Function prototype</b>	void usart_parity_config(uint32_t usart_periph, uint32_t paritycfg);
<b>Function descriptions</b>	configure USART parity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral



<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>paritycfg</b>	configure USART parity
<i>USART_PM_NONE</i>	no parity
<i>USART_PM_ODD</i>	odd parity
<i>USART_PM_EVEN</i>	even parity
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1339. Function usart\_word\_length\_set**

<b>Function name</b>	usart_word_length_set
<b>Function prototype</b>	void usart_word_length_set(uint32_t usart_periph, uint32_t wlen);
<b>Function descriptions</b>	configure USART word length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>wlen</b>	USART word length configure
<i>USART_WL_8BIT</i>	8 bits
<i>USART_WL_9BIT</i>	9 bits
<i>USART_WL_7BIT</i>	7 bits
<i>USART_WL_10BIT</i>	10 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1340. Function usart\_stop\_bit\_set**

<b>Function name</b>	usart_stop_bit_set
<b>Function prototype</b>	void usart_stop_bit_set(uint32_t usart_periph, uint32_t stblen);
<b>Function descriptions</b>	configure USART stop bit length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>stblen</b>	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
<i>USART_STB_1_5BIT</i>	1.5 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1341. Function usart\_enable**

<b>Function name</b>	usart_enable
<b>Function prototype</b>	void usart_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable USART
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2

<i>UARTx</i>	<i>x</i> = 3, 4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 */
usart_enable(USART0);
```

### usart\_disable

The description of usart\_disable is shown as below:

**Table 3-1342. Function usart\_disable**

<b>Function name</b>	usart_disable
<b>Function prototype</b>	void usart_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable USART
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	<i>x</i> = 0, 1, 2
<i>UARTx</i>	<i>x</i> = 3, 4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 */
usart_disable(USART0);
```

### usart\_transmit\_config

The description of usart\_transmit\_config is shown as below:

**Table 3-1343. Function usart\_transmit\_config**

<b>Function name</b>	usart_transmit_config
<b>Function prototype</b>	void usart_transmit_config(uint32_t usart_periph, uint32_t txconfig);
<b>Function descriptions</b>	configure USART transmitter
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
Input parameter{in}	
<b>txconfig</b>	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 3-1344. Function usart\_receive\_config**

<b>Function name</b>	usart_receive_config
<b>Function prototype</b>	void usart_receive_config(uint32_t usart_periph, uint32_t rxconfig);
<b>Function descriptions</b>	configure USART receiver
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
Input parameter{in}	
<b>rxconfig</b>	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 3-1345. Function usart\_data\_first\_config**

<b>Function name</b>	usart_data_first_config
<b>Function prototype</b>	void usart_data_first_config(uint32_t usart_periph, uint32_t msbf);
<b>Function descriptions</b>	data is transmitted/received with the LSB/MSB first
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>msbf</b>	LSB/MSB
USART_MSBF_LSB	LSB first
USART_MSBF_MSB	MSB first
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1346. Function usart\_invert\_config**

<b>Function name</b>	usart_invert_config
<b>Function prototype</b>	void usart_invert_config(uint32_t usart_periph, usart_invert_enum invertpara);
<b>Function descriptions</b>	configure USART inverted
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>invertpara</b>	refer to <a href="#">Table 3-1335. Enum usart_invert_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 inversion */
```

```
usart_invert_config(USART0, USART_DINV_ENABLE);
```

### usart\_overshoot\_enable

The description of usart\_overshoot\_enable is shown as below:

**Table 3-1347. Function usart\_overshoot\_enable**

<b>Function name</b>	usart_overshoot_enable
<b>Function prototype</b>	void usart_overshoot_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable the USART overshoot function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 overshoot */
```

```
usart_overshoot_enable(USART0);
```

### usart\_overshoot\_disable

The description of usart\_overshoot\_disable is shown as below:

**Table 3-1348. Function usart\_overshoot\_disable**

<b>Function name</b>	usart_overshoot_disable
----------------------	-------------------------

<b>Function prototype</b>	void usart_overrun_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable the USART overrun function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 overrun */
usart_overrun_disable(USART0);
```

### usart\_oversample\_config

The description of usart\_oversample\_config is shown as below:

**Table 3-1349. Function usart\_oversample\_config**

<b>Function name</b>	usart_oversample_config
<b>Function prototype</b>	void usart_oversample_config(uint32_t usart_periph, uint32_t oversamp);
<b>Function descriptions</b>	configure the USART oversample mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>oversamp</b>	oversample value
<i>USART_OVSMOD_8</i>	oversampling by 8
<i>USART_OVSMOD_16</i>	oversampling by 16
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure 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 3-1350. Function usart\_sample\_bit\_config**

<b>Function name</b>	usart_sample_bit_config
<b>Function prototype</b>	void usart_sample_bit_config(uint32_t usart_periph, uint32_t osb);
<b>Function descriptions</b>	configure the sample bit method
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>osb</b>	sample bit
<i>USART_OSB_1BIT</i>	1 bit
<i>USART_OSB_3BIT</i>	3 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1351. Function usart\_receiver\_timeout\_enable**

<b>Function name</b>	usart_receiver_timeout_enable
<b>Function prototype</b>	void usart_receiver_timeout_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable receiver timeout
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-



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 3-1352. Function usart\_receiver\_timeout\_disable**

<b>Function name</b>	usart_receiver_timeout_disable
<b>Function prototype</b>	void usart_receiver_timeout_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable receiver timeout
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1353. Function usart\_receiver\_timeout\_threshold\_config**

<b>Function name</b>	usart_receiver_timeout_threshold_config
<b>Function prototype</b>	void usart_receiver_timeout_threshold_config(uint32_t usart_periph, uint32_t rtimeout);
<b>Function descriptions</b>	configure receiver timeout threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Input parameter{in}</b>	
<b>rtimeout</b>	receiver timeout (0x00000000-0x00FFFFFF)
<b>Output parameter{out}</b>	

-	-
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 3-1354. Function usart\_data\_transmit**

<b>Function name</b>	usart_data_transmit
<b>Function prototype</b>	void usart_data_transmit(uint32_t usart_periph, uint16_t data);
<b>Function descriptions</b>	USART transmit data function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>data</b>	data of transmission (0x0000-0x01FF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* USART0 transmit data */
```

```
usart_data_transmit(USART0, 0xAA);
```

### usart\_data\_receive

The description of usart\_data\_receive is shown as below:

**Table 3-1355. Function usart\_data\_receive**

<b>Function name</b>	usart_data_receive
<b>Function prototype</b>	uint16_t usart_data_receive(uint32_t usart_periph);
<b>Function descriptions</b>	USART receive data function
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
Output parameter{out}	
-	-
Return value	
<b>uint16_t</b>	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 3-1356. Function usart\_command\_enable**

<b>Function name</b>	usart_command_enable
<b>Function prototype</b>	void usart_command_enable(uint32_t usart_periph, uint32_t cmdtype);
<b>Function descriptions</b>	enable USART command
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
Input parameter{in}	
<b>cmdtype</b>	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\_0\_match\_mode\_enable

The description of usart\_address\_0\_match\_mode\_enable is shown as below:

**Table 3-1357. Function usart\_address\_0\_match\_mode\_enable**

<b>Function name</b>	usart_address_0_match_mode_enable
<b>Function prototype</b>	void usart_address_0_match_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable address 0 match mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable address 0 match mode */
```

```
usart_address_0_match_mode_enable (USART0);
```

### usart\_address\_0\_match\_mode\_disable

The description of usart\_address\_0\_match\_mode\_disable is shown as below:

**Table 3-1358. Function usart\_address\_0\_match\_mode\_disable**

<b>Function name</b>	usart_address_0_match_mode_disable
<b>Function prototype</b>	void usart_address_0_match_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable address 0 match mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable address 0 match mode */
```

```
usart_address_0_match_mode_disable(USART0);
```

### usart\_address\_1\_match\_mode\_enable

The description of usart\_address\_1\_match\_mode\_enable is shown as below:

**Table 3-1359. Function usart\_address\_1\_match\_mode\_enable**

<b>Function name</b>	usart_address_1_match_mode_enable
<b>Function prototype</b>	void usart_address_1_match_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable address 1 match mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable address 1 match mode */
```

```
usart_address_1_match_mode_enable (USART0);
```

### usart\_address\_1\_match\_mode\_disable

The description of usart\_address\_1\_match\_mode\_disable is shown as below:

**Table 3-1360. Function usart\_address\_1\_match\_mode\_disable**

<b>Function name</b>	usart_address_1_match_mode_disable
<b>Function prototype</b>	void usart_address_1_match_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable address 1 match mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable address 1 match mode */
```

```
usart_address_1_match_mode_disable(USART0);
```

### usart\_address\_0\_config

The description of usart\_address\_0\_config is shown as below:

**Table 3-1361. Function usart\_address\_0\_config**

Function name	usart_address_0_config
Function prototype	void usart_address_0_config(uint32_t usart_periph, uint8_t addr);
Function descriptions	configure address 0 of the USART
Precondition	-
The called functions	-
Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
Input parameter{in}	
<b>addr</b>	address of USART (0x00-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address 0 of the USART0 */
```

```
usart_address_0_config(USART0, 0x00);
```

### usart\_address\_1\_config

The description of usart\_address\_1\_config is shown as below:

**Table 3-1362. Function usart\_address\_1\_config**

Function name	usart_address_1_config
Function prototype	void usart_address_1_config(uint32_t usart_periph, uint8_t addr);
Function descriptions	configure address 1 of the USART
Precondition	-

The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
Input parameter{in}	
addr	address of USART (0x00-0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure address 1 of the USART0 */
```

```
usart_address_1_config(USART0, 0x00);
```

### usart\_address\_0\_detection\_mode\_config

The description of usart\_address\_0\_detection\_mode\_config is shown as below:

**Table 3-1363. Function usart\_address\_0\_detection\_mode\_config**

Function name	usart_address_0_detection_mode_config
Function prototype	void usart_address_0_detection_mode_config(uint32_t usart_periph, uint32_t addmod);
Function descriptions	configure address 0 detection mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
Input parameter{in}	
addmod	address detection mode
USART_ADDM_4BIT	4 bits
USART_ADDM_FULLBIT	full bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*configure address 0 detection mode */
```

```
usart_address_0_detection_mode_config(USART0, USART_ADDM_4BIT);
```

### usart\_address\_1\_detection\_mode\_config

The description of usart\_address\_1\_detection\_mode\_config is shown as below:

**Table 3-1364. Function usart\_address\_1\_detection\_mode\_config**

<b>Function name</b>	usart_address_1_detection_mode_config
<b>Function prototype</b>	void usart_address_1_detection_mode_config(uint32_t usart_periph, uint32_t addmod);
<b>Function descriptions</b>	configure address 1 detection mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>addmod</b>	address detection mode
USART_ADDM_4BIT	4 bits
USART_ADDM_FULLBIT	full bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*configure address 1 detection mode */
```

```
usart_address_1_detection_mode_config(USART0, USART_ADDM_4BIT);
```

### usart\_mute\_mode\_enable

The description of usart\_mute\_mode\_enable is shown as below:

**Table 3-1365. Function usart\_mute\_mode\_enable**

<b>Function name</b>	usart_mute_mode_enable
<b>Function prototype</b>	void usart_mute_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable mute mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2



<i>UARTx</i>	<i>x</i> = 3, 4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 receiver in mute mode */
```

```
usart_mute_mode_enable(USART0);
```

### usart\_mute\_mode\_disable

The description of usart\_mute\_mode\_disable is shown as below:

**Table 3-1366. Function usart\_mute\_mode\_disable**

<b>Function name</b>	usart_mute_mode_disable
<b>Function prototype</b>	void usart_mute_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable mute mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	<i>x</i> = 0, 1, 2
<i>UARTx</i>	<i>x</i> = 3, 4
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 3-1367. Function usart\_mute\_mode\_wakeup\_config**

<b>Function name</b>	usart_mute_mode_wakeup_config
<b>Function prototype</b>	void usart_mute_mode_wakeup_config(uint32_t usart_periph, uint32_t wmethod);
<b>Function descriptions</b>	configure wakeup method in mute mode
<b>Precondition</b>	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>wmethod</b>	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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1368. Function usart\_lin\_mode\_enable**

<b>Function name</b>	usart_lin_mode_enable
<b>Function prototype</b>	void usart_lin_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable LIN mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 LIN mode */
usart_lin_mode_enable(USART0);
```

### usart\_lin\_mode\_disable

The description of usart\_lin\_mode\_disable is shown as below:

Table 3-1369. Function `usart_lin_mode_disable`

Function name	<code>usart_lin_mode_disable</code>
Function prototype	<code>void usart_lin_mode_disable(uint32_t usart_periph);</code>
Function descriptions	disable LIN mode
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	usart peripheral
<code>USARTx</code>	x = 0, 1, 2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 LIN mode */
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 3-1370. Function `usart_lin_break_dection_length_config`

Function name	<code>usart_lin_break_dection_length_config</code>
Function prototype	<code>void usart_lin_break_dection_length_config(uint32_t usart_periph, uint32_t lflen);</code>
Function descriptions	LIN break detection length
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	usart peripheral
<code>USARTx</code>	x = 0, 1, 2
Input parameter{in}	
<code>lflen</code>	two methods be used to enter or exit the mute mode
<code>USART_LBLEN_10B</code>	10 bits
<code>USART_LBLEN_11B</code>	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 3-1371. Function usart\_halfduplex\_enable**

<b>Function name</b>	usart_halfduplex_enable
<b>Function prototype</b>	void usart_halfduplex_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable half-duplex mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 half duplex mode */
```

```
usart_halfduplex_enable(USART0);
```

## usart\_halfduplex\_disable

The description of usart\_halfduplex\_disable is shown as below:

**Table 3-1372. Function usart\_halfduplex\_disable**

<b>Function name</b>	usart_halfduplex_disable
<b>Function prototype</b>	void usart_halfduplex_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable half-duplex mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 half duplex mode */
```

```
usart_halfduplex_disable(USART0);
```

### usart\_clock\_enable

The description of usart\_clock\_enable is shown as below:

**Table 3-1373. Function usart\_clock\_enable**

<b>Function name</b>	usart_clock_enable
<b>Function prototype</b>	void usart_clock_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable clock */
```

```
usart_clock_enable(USART0);
```

### usart\_clock\_disable

The description of usart\_clock\_disable is shown as below:

**Table 3-1374. Function usart\_clock\_disable**

<b>Function name</b>	usart_clock_disable
<b>Function prototype</b>	void usart_clock_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable clock */
```

```
usart_clock_disable(USART0);
```

### usart\_synchronous\_clock\_config

The description of usart\_synchronous\_clock\_config is shown as below:

**Table 3-1375. Function usart\_synchronous\_clock\_config**

<b>Function name</b>	usart_synchronous_clock_config
<b>Function prototype</b>	void usart_synchronous_clock_config(uint32_t usart_periph, uint32_t clen, uint32_t cph, uint32_t cpl);
<b>Function descriptions</b>	configure USART synchronous mode parameters
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2
<b>Input parameter{in}</b>	
<b>clen</b>	last bit clock pulse
USART_CLEN_NONE	clock pulse of the last data bit (MSB) is not output to the CK pin
USART_CLEN_EN	clock pulse of the last data bit (MSB) is output to the CK pin
<b>Input parameter{in}</b>	
<b>cph</b>	clock phase
USART_CPH_1CK	first clock transition is the first data capture edge
USART_CPH_2CK	second clock transition is the first data capture edge
<b>Input parameter{in}</b>	
<b>cpl</b>	clock polarity
USART_CPL_LOW	steady low value on CK pin
USART_CPL_HIGH	steady high value on CK pin
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1376. Function usart\_guard\_time\_config

<b>Function name</b>	usart_guard_time_config
<b>Function prototype</b>	void usart_guard_time_config(uint32_t usart_periph,uint32_t guat);
<b>Function descriptions</b>	configure guard time value in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Input parameter{in}</b>	
<b>guat</b>	guard time value (0x00000000-0x000000FF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1377. Function usart\_smartcard\_mode\_enable

<b>Function name</b>	usart_smartcard_mode_enable
<b>Function prototype</b>	void usart_smartcard_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 smartcard mode */
usart_smartcard_mode_enable(USART0);
```

## usart\_smartcard\_mode\_disable

The description of usart\_smartcard\_mode\_disable is shown as below:

**Table 3-1378. Function usart\_smartcard\_mode\_disable**

<b>Function name</b>	usart_smartcard_mode_disable
<b>Function prototype</b>	void usart_smartcard_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 smartcard mode */
usart_smartcard_mode_disable(USART0);
```

## usart\_smartcard\_mode\_nack\_enable

The description of usart\_smartcard\_mode\_nack\_enable is shown as below:

**Table 3-1379. Function usart\_smartcard\_mode\_nack\_enable**

<b>Function name</b>	usart_smartcard_mode_nack_enable
<b>Function prototype</b>	void usart_smartcard_mode_nack_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable NACK in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1380. Function usart\_smartcard\_mode\_nack\_disable**

<b>Function name</b>	usart_smartcard_mode_nack_disable
<b>Function prototype</b>	void usart_smartcard_mode_nack_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable NACK in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1381. Function usart\_smartcard\_mode\_early\_nack\_enable**

<b>Function name</b>	usart_smartcard_mode_early_nack_enable
<b>Function prototype</b>	void usart_smartcard_mode_early_nack_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable early NACK in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1382. Function usart\_smartcard\_mode\_early\_nack\_disable**

<b>Function name</b>	usart_smartcard_mode_early_nack_disable
<b>Function prototype</b>	void usart_smartcard_mode_early_nack_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable early NACK in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1383. Function usart\_smartcard\_autoretry\_config**

<b>Function name</b>	usart_smartcard_autoretry_config
<b>Function prototype</b>	void usart_smartcard_autoretry_config(uint32_t usart_periph, uint32_t scrtnum);
<b>Function descriptions</b>	configure smartcard auto-retry number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Input parameter{in}</b>	
<b>scrtnum</b>	smartcard auto-retry number (0x00000000-0x00000007)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1384. Function usart\_block\_length\_config**

<b>Function name</b>	usart_block_length_config
<b>Function prototype</b>	void usart_block_length_config(uint32_t usart_periph, uint32_t bl);
<b>Function descriptions</b>	configure block length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Input parameter{in}</b>	
<b>bl</b>	block length(0x00000000-0x000000FF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1385. Function usart\_irda\_mode\_enable**

<b>Function name</b>	usart_irda_mode_enable
<b>Function prototype</b>	void usart_irda_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable IrDA mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

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 3-1386. Function usart\_irda\_mode\_disable**

<b>Function name</b>	usart_irda_mode_disable
<b>Function prototype</b>	void usart_irda_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable IrDA mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 IrDA mode */
usart_irda_mode_disable(USART0);
```

### usart\_prescaler\_config

The description of usart\_prescaler\_config is shown as below:

**Table 3-1387. Function usart\_prescaler\_config**

<b>Function name</b>	usart_prescaler_config
<b>Function prototype</b>	void usart_prescaler_config(uint32_t usart_periph, uint8_t psc);
<b>Function descriptions</b>	configure the peripheral clock prescaler in USART IrDA low-power or smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Input parameter{in}</b>	

<b>psc</b>	clock prescaler (0x00-0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the USART0 peripheral clock prescaler in USART IrDA low-power mode */
usart_prescaler_config(USART0, 0x00);
```

### usart\_irda\_lowpower\_config

The description of usart\_irda\_lowpower\_config is shown as below:

**Table 3-1388. Function usart\_irda\_lowpower\_config**

<b>Function name</b>	usart_irda_lowpower_config
<b>Function prototype</b>	void usart_irda_lowpower_config(uint32_t usart_periph, uint32_t irlp);
<b>Function descriptions</b>	configure IrDA low-power
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2
<b>Input parameter{in}</b>	
<b>irlp</b>	IrDA low-power or normal
USART_IRLP_LOW	low-power
USART_IRLP_NORMAL	normal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1389. Function usart\_hardware\_flow\_rts\_config**

<b>Function name</b>	usart_hardware_flow_rts_config
----------------------	--------------------------------

<b>Function prototype</b>	void usart_hardware_flow_rts_config(uint32_t usart_periph, uint32_t rtsconfig);
<b>Function descriptions</b>	configure hardware flow control RTS
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>rtsconfig</b>	enable or disable RTS
USART_RTS_ENABLE	enable RTS
USART_RTS_DISABLE	disable RTS
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1390. Function usart\_hardware\_flow\_cts\_config**

<b>Function name</b>	usart_hardware_flow_cts_config
<b>Function prototype</b>	void usart_hardware_flow_cts_config(uint32_t usart_periph, uint32_t ctsconfig);
<b>Function descriptions</b>	configure hardware flow control CTS
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>ctsconfig</b>	enable or disable CTS
USART_CTS_ENABLE	enable CTS
USART_CTS_DISABLE	disable CTS

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 hardware flow control CTS */
```

```
usart_hardware_flow_cts_config(USART0, USART_CTS_ENABLE);
```

### usart\_hardware\_flow\_coherence\_config

The description of usart\_hardware\_flow\_coherence\_config is shown as below:

**Table 3-1391. 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}	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
Input parameter{in}	
<b>hcm</b>	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 3-1392. Function usart\_rs485\_driver\_enable**

Function name	usart_rs485_driver_enable
---------------	---------------------------

<b>Function prototype</b>	void usart_rs485_driver_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable USART RS485 driver
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1393. Function usart\_rs485\_driver\_disable**

<b>Function name</b>	usart_rs485_driver_disable
<b>Function prototype</b>	void usart_rs485_driver_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable USART RS485 driver
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 RS485 driver */
usart_rs485_driver_disable(USART0);
```

### usart\_driver\_assertime\_config

The description of usart\_driver\_assertime\_config is shown as below:



Table 3-1394. Function `usart_driver_assertime_config`

Function name	<code>usart_driver_assertime_config</code>
Function prototype	<code>void usart_driver_assertime_config(uint32_t usart_periph, uint32_t deatime);</code>
Function descriptions	configure driver enable assertion time
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	usart peripheral
<code>USARTx</code>	x = 0, 1, 2
<code>UARTx</code>	x = 3, 4
Input parameter{in}	
<code>deatime</code>	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 3-1395. Function `usart_driver_deassertime_config`

Function name	<code>usart_driver_deassertime_config</code>
Function prototype	<code>void usart_driver_deassertime_config(uint32_t usart_periph, uint32_t dedtime);</code>
Function descriptions	configure driver enable de-assertion time
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	usart peripheral
<code>USARTx</code>	x = 0, 1, 2
<code>UARTx</code>	x = 3, 4
Input parameter{in}	
<code>dedtime</code>	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 3-1396. 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}	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
Input parameter{in}	
<b>dep</b>	DE signal
USART_DEP_HIGH	DE signal is active high
USART_DEP_LOW	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 3-1397. Function usart\_dma\_receive\_config**

Function name	usart_dma_receive_config
Function prototype	void usart_dma_receive_config(uint32_t usart_periph, uint32_t dmactmd);
Function descriptions	configure USART DMA for reception
Precondition	-
The called functions	-
Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral

USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>dmacmd</b>	USART DMA mode
USART_RECEIVE_DMA_ENABLE	USART enable DMA for reception
USART_RECEIVE_DMA_DISABLE	USART disable DMA for reception
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* USART0 DMA enable for reception */
```

```
usart_dma_receive_config(USART0, USART_RECEIVE_DMA_ENABLE);
```

### usart\_dma\_transmit\_config

The description of usart\_dma\_transmit\_config is shown as below:

**Table 3-1398. Function usart\_dma\_transmit\_config**

<b>Function name</b>	usart_dma_transmit_config
<b>Function prototype</b>	void usart_dma_transmit_config(uint32_t usart_periph, uint32_t dmacmd);
<b>Function descriptions</b>	configure USART DMA transmission
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>dmacmd</b>	USART DMA mode
USART_TRANSMIT_DMA_ENABLE	USART enable DMA for transmission
USART_TRANSMIT_DMA_DISABLE	USART disable DMA for transmission
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* USART0 DMA disable for transmission */
```

```
usart_dma_transmit_config (USART0, USART_TRANSMIT_DMA_DISABLE);
```

### usart\_reception\_error\_dma\_disable

The description of usart\_reception\_error\_dma\_disable is shown as below:

**Table 3-1399. Function usart\_reception\_error\_dma\_disable**

<b>Function name</b>	usart_reception_error_dma_disable
<b>Function prototype</b>	void usart_reception_error_dma_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable DMA on reception error
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1400. Function usart\_reception\_error\_dma\_enable**

<b>Function name</b>	usart_reception_error_dma_enable
<b>Function prototype</b>	void usart_reception_error_dma_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable DMA on reception error
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

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 3-1401. Function usart\_wakeup\_enable**

<b>Function name</b>	usart_wakeup_enable
<b>Function prototype</b>	void usart_wakeup_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable USART to wakeup the mcu from deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 wake up */

usart_wakeup_enable(USART0);
```

## usart\_wakeup\_disable

The description of usart\_wakeup\_disable is shown as below:

**Table 3-1402. Function usart\_wakeup\_disable**

<b>Function name</b>	usart_wakeup_disable
<b>Function prototype</b>	void usart_wakeup_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable USART to wakeup the mcu from deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* disable USART0 wake up */
```

```
usart_wakeup_disable(USART0);
```

### usart\_wakeup\_mode\_config

The description of usart\_wakeup\_mode\_config is shown as below:

**Table 3-1403. 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
USART_WUM_ADDR	WUF active on address match
USART_WUM_START B	WUF active on start bit
USART_WUM_RBNE	WUF active on RBNE
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 wake up mode */
```

```
usart_wakeup_mode_config(USART0, USART_WUM_ADDR);
```

### usart\_fifo\_enable

The description of usart\_fifo\_enable is shown as below:

**Table 3-1404. Function usart\_fifo\_enable**

Function name	usart_fifo_enable
Function prototype	void usart_fifo_enable(uint32_t usart_periph);
Function descriptions	enable FIFO

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FIFO */
```

```
usart_fifo_enable (USART0);
```

### usart\_fifo\_disable

The description of usart\_fifo\_disable is shown as below:

**Table 3-1405. Function usart\_fifo\_disable**

<b>Function name</b>	usart_fifo_disable
<b>Function prototype</b>	void usart_fifo_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable FIFO
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FIFO */
```

```
usart_fifo_disable(USART0);
```

### usart\_transmit\_fifo\_threshold\_config

The description of usart\_transmit\_fifo\_threshold\_config is shown as below:

Table 3-1406. Function `usart_transmit_fifo_threshold_config`

<b>Function name</b>	<code>usart_transmit_fifo_threshold_config</code>
<b>Function prototype</b>	<code>void usart_transmit_fifo_threshold_config(uint32_t usart_periph, uint32_t txthreshold);</code>
<b>Function descriptions</b>	configure transmit FIFO threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>txthreshold</b>	transmit FIFO threshold
<i>USART_TFTCFG_THRESHOLD_1_8</i>	transmit FIFO reaches 1/8 of its depth
<i>USART_TFTCFG_THRESHOLD_1_4</i>	transmit FIFO reaches 1/4 of its depth
<i>USART_TFTCFG_THRESHOLD_1_2</i>	transmit FIFO reaches 1/8 of its depth
<i>USART_TFTCFG_THRESHOLD_3_4</i>	transmit FIFO reaches 3/4 of its depth
<i>USART_TFTCFG_THRESHOLD_7_8</i>	transmit FIFO reaches 7/8 of its depth
<i>USART_TFTCFG_THRESHOLD_EMPTY</i>	transmit FIFO becomes empty
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure transmit FIFO threshold as empty */
```

```
usart_transmit_fifo_threshold_config (USART0, USART_TFTCFG_THRESHOLD_EMPTY);
```

### **usart\_receive\_fifo\_threshold\_config**

The description of `usart_receive_fifo_threshold_config` is shown as below:

Table 3-1407. Function `usart_receive_fifo_threshold_config`

<b>Function name</b>	<code>usart_receive_fifo_threshold_config</code>
----------------------	--



<b>Function prototype</b>	void usart_receive_fifo_threshold_config(uint32_t usart_periph, uint32_t rxthreshold);
<b>Function descriptions</b>	configure receive FIFO threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2
UARTx	x = 3, 4
<b>Input parameter{in}</b>	
<b>rxthreshold</b>	receive FIFO threshold
USART_RFTCFG_THRESHOLD_1_8	receive FIFO reaches 1/8 of its depth
USART_RFTCFG_THRESHOLD_1_4	receive FIFO reaches 1/4 of its depth
USART_RFTCFG_THRESHOLD_1_2	receive FIFO reaches 1/2 of its depth
USART_RFTCFG_THRESHOLD_3_4	receive FIFO reaches 3/4 of its depth
USART_RFTCFG_THRESHOLD_7_8	receive FIFO reaches 7/8 of its depth
USART_RFTCFG_THRESHOLD_FULL	receive FIFO becomes full
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure receiveFIFO threshold as full */
```

```
usart_receive_fifo_threshold_config (USART0, USART_RFTCFG_THRESHOLD_FULL);
```

### usart\_receive\_fifo\_counter\_number

The description of usart\_receive\_fifo\_counter\_number is shown as below:

**Table 3-1408. Function usart\_receive\_fifo\_counter\_number**

<b>Function name</b>	usart_receive_fifo_counter_number
<b>Function prototype</b>	uint8_t usart_receive_fifo_counter_number(uint32_t usart_periph);
<b>Function descriptions</b>	read receive FIFO counter number

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint8_t</b>	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 3-1409. Function usart\_flag\_get**

<b>Function name</b>	usart_flag_get
<b>Function prototype</b>	FlagStatus usart_flag_get(uint32_t usart_periph, usart_flag_enum flag);
<b>Function descriptions</b>	get flag in STAT/CHC/FCS register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2
<i>UARTx</i>	x = 3, 4
<b>Input parameter{in}</b>	
<b>flag</b>	USART flags, refer to <a href="#">Table 3-1332. Enum usart_flag_enum</a> only one among these parameters can be selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-1410. Function usart\_flag\_clear**

<b>Function name</b>	usart_flag_clear
<b>Function prototype</b>	void usart_flag_clear(uint32_t usart_periph, usart_flag_enum flag);
<b>Function descriptions</b>	clear flag in STAT register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>flag</b>	USART flags, refer to <a href="#">Table 3-1332. Enum usart_flag_enum</a> only one among these parameters can be selected
<i>USART_FLAG_PERR</i>	parity error flag
<i>USART_FLAG_FERR</i>	frame error flag
<i>USART_FLAG_NERR</i>	noise detected flag
<i>USART_FLAG_ORER R</i>	overrun error flag
<i>USART_FLAG_IDLE</i>	idle line detected flag
<i>USART_FLAG_TC</i>	transmission complete flag
<i>USART_FLAG_LBD</i>	LIN break detected flag
<i>USART_FLAG_CTSF</i>	CTS change flag
<i>USART_FLAG_RT</i>	receiver timeout flag
<i>USART_FLAG_EB</i>	end of block flag
<i>USART_FLAG_AM0</i>	address 0 match flag
<i>USART_FLAG_AM1</i>	address 1 match flag
<i>USART_FLAG_WU</i>	wakeup from deep-sleep mode flag
<i>USART_FLAG_EPERR</i>	early parity error flag
<i>USART_FLAG_TFE</i>	transmit FIFO empty flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1411. Function usart\_interrupt\_enable**

<b>Function name</b>	usart_interrupt_enable
<b>Function prototype</b>	void usart_interrupt_enable(uint32_t usart_periph, usart_interrupt_enum interrupt);
<b>Function descriptions</b>	enable USART interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>interrupt</b>	interrupt type, refer to <a href="#">Table 3-1334. Enum usart_interrupt_enum</a> only one among these parameters can be selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1412. Function usart\_interrupt\_disable**

<b>Function name</b>	usart_interrupt_disable
<b>Function prototype</b>	void usart_interrupt_disable(uint32_t usart_periph, usart_interrupt_enum interrupt);
<b>Function descriptions</b>	disable USART interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>interrupt</b>	interrupt type, refer to <a href="#">Table 3-1334. Enum usart_interrupt_enum</a> only one among these parameters can be selected
<b>Output parameter{out}</b>	

-	-
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 3-1413. Function usart\_interrupt\_flag\_get**

<b>Function name</b>	usart_interrupt_flag_get
<b>Function prototype</b>	FlagStatus usart_interrupt_flag_get(uint32_t usart_periph, usart_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	get USART interrupt and flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>int_flag</b>	USART interrupt flag, refer to <a href="#">Table 3-1333. Enum usart_interrupt_flag_enum</a> , only one among these parameters can be selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-1414. Function usart\_interrupt\_flag\_clear**

<b>Function name</b>	usart_interrupt_flag_clear
<b>Function prototype</b>	void usart_interrupt_flag_clear(uint32_t usart_periph,

	usart_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	clear USART interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<b>USARTx</b>	x=0,1,2
<b>Input parameter{in}</b>	
<b>int_flag</b>	USART interrupt flag, refer to <a href="#">Table 3-1333. Enum usart_interrupt_flag_enum</a> , only one among these parameters can be selected
USART_INT_FLAG_PERRR	parity error flag
USART_INT_FLAG_ERRFR	frame error flag
USART_INT_FLAG_ERRNR	noise detected flag
USART_INT_FLAG_RBNE_ORERR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_ERR_ORERR	error interrupt and overrun error
USART_INT_FLAG_IDLE	idle line detected flag
USART_INT_FLAG_TC	transmission complete flag
USART_INT_FLAG_LBD	LIN break detected flag
USART_INT_FLAG_CTS	CTS change flag
USART_INT_FLAG_RRT	receiver timeout flag
USART_INT_FLAG_EB	end of block flag
USART_INT_FLAG_AM0	address 0 match flag
USART_INT_FLAG_AM1	address 1 match flag
USART_INT_FLAG_WU	wakeup from deep-sleep mode flag
USART_INT_RFT	receive FIFO threshold interrupt
USART_INT_FLAG_RFF	receive FIFO full interrupt and flag
USART_INT_FLAG_TFE	transmit FIFO empty interrupt and flag
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* clear the USART0 interrupt flag */
```

```
usart_interrupt_flag_clear(USART0, USART_INT_FLAG_TC);
```

## 3.35. VREF

The precision internal reference is used to provide reference voltage for ADC / DAC, or used by off-chip circuit connecting to VREFP pin. The VREF registers are listed in chapter [3.35.1](#), the VREF firmware functions are introduced in chapter [3.35.2](#).

### 3.35.1. Descriptions of Peripheral registers

VREF registers are listed in the table shown as below:

**Table 3-1415. VREF Registers**

Registers	Descriptions
VREF_CS	VREF control and status register
VREF_CALIB	VREF calibration register

### 3.35.2. Descriptions of Peripheral functions

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

**Table 3-1416. VREF firmware function**

Function name	Function description
vref_deinit	deinitialize the VREF
vref_enable	enable VREF
vref_disable	disable VREF
vref_high_impedance_mode_enable	enable VREF high impedance mode
vref_high_impedance_mode_disable	disable VREF high impedance mode
vref_status_get	get the status of VREF
vref_voltage_select	select the VREF voltage reference
vref_calib_value_set	set the calibration value of VREF
vref_calib_value_get	get the calibration value of VREF

#### vref\_deinit

The description of vref\_deinit is shown as below:

**Table 3-1417. Function vref\_deinit**

Function name	vref_deinit
---------------	-------------

<b>Function prototype</b>	void vref_deinit(void);
<b>Function descriptions</b>	deinitialize the VREF
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize the VREF */
vref_deinit();
```

### vref\_enable

The description of vref\_enable is shown as below:

**Table 3-1418. Function vref\_enable**

<b>Function name</b>	vref_enable
<b>Function prototype</b>	void vref_enable(void);
<b>Function descriptions</b>	enable VREF
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable VREF */
vref_enable();
```

### vref\_disable

The description of vref\_disable is shown as below:

**Table 3-1419. Function vref\_disable**

<b>Function name</b>	vref_disable
<b>Function prototype</b>	void vref_disable(void);



<b>Function descriptions</b>	disable VREF
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable VREF */
vref_disable();
```

### vref\_high\_impedance\_mode\_enable

The description of vref\_high\_impedance\_mode\_enable is shown as below:

**Table 3-1420. Function vref\_high\_impedance\_mode\_enable**

<b>Function name</b>	vref_high_impedance_mode_enable
<b>Function prototype</b>	void vref_high_impedance_mode_enable(void);
<b>Function descriptions</b>	enable VREF high impedance mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable VREF high impedance mode */
vref_high_impedance_mode_enable();
```

### vref\_high\_impedance\_mode\_disable

The description of vref\_high\_impedance\_mode\_disable is shown as below:

**Table 3-1421. Function vref\_high\_impedance\_mode\_disable**

<b>Function name</b>	vref_high_impedance_mode_disable
<b>Function prototype</b>	void vref_high_impedance_mode_disable(void);
<b>Function descriptions</b>	disable VREF high impedance mode

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

Example:

```
/* disable VREF high impedance mode */
vref_high_impedance_mode_disable();
```

### vref\_status\_get

The description of vref\_status\_get is shown as below:

**Table 3-1422. Function vref\_status\_get**

Function name	vref_status_get
Function prototype	FlagStatus vref_status_get(void);
Function descriptions	get the status of VREF
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the status of VREF */
FlagStatus status;
status = vref_status_get();
```

### vref\_voltage\_select

The description of vref\_voltage\_select is shown as below:

**Table 3-1423. Function vref\_voltage\_select**

Function name	vref_voltage_select
Function prototype	void vref_voltage_select(uint32_t vref_voltage);
Function descriptions	select the VREF voltage reference

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>vref_voltage</b>	VREF voltage reference select
VREF_VOLTAGE_SEL_2_048V	VREF voltage reference select 2.048 V
VREF_VOLTAGE_SEL_2_5V	VREF voltage reference select 2.5 V
VREF_VOLTAGE_SEL_2_9V	VREF voltage reference select 2.9 V
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select 2.5V as the VREF voltage reference */
```

```
vref_voltage_select(VREF_VOLTAGE_SEL_2_5V);
```

### vref\_calib\_value\_set

The description of vref\_calib\_value\_set is shown as below:

**Table 3-1424. Function vref\_calib\_value\_set**

<b>Function name</b>	vref_calib_value_set
<b>Function prototype</b>	void vref_calib_value_set(uint8_t value);
<b>Function descriptions</b>	set the calibration value of VREF
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>value</b>	calibration value (0x00 - 0x3F)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the calibration value of VREF */
```

```
vref_calib_value_set(0x0A);
```

### vref\_calib\_value\_get

The description of vref\_calib\_value\_get is shown as below:

Table 3-1425. Function vref\_calib\_value\_get

Function name	vref_calib_value_get
Function prototype	uint8_t vref_calib_value_get(void);
Function descriptions	get the calibration value of VREF
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint8_t	calibration value (0x00 - 0x3F)

Example:

```
/* get the calibration value of VREF */
uint8_t cal_val;
cal_val = vref_calib_value_get();
```

## 3.36. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. The WWDGT registers are listed in chapter [3.36.1](#), the WWDGT firmware functions are introduced in chapter [3.36.2](#).

### 3.36.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

Table 3-1426. WWDGT Registers

Registers	Descriptions
WWDGT_CTL	WWDGT control register
WWDGT_CFG	WWDGT configuration register
WWDGT_STAT	WWDGT status register

### 3.36.2. Descriptions of Peripheral functions

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

Table 3-1427. WWDGT firmware function

Function name	Function description
wwdgt_deinit	reset the window watchdog timer configuration
wwdgt_enable	start the window watchdog timer counter

Function name	Function description
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 3-1428. 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 3-1429. 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 3-1430. Function wwdgt\_counter\_update**

<b>Function name</b>	wwdgt_counter_update
<b>Function prototype</b>	void wwdgt_counter_update(uint16_t counter_value);
<b>Function descriptions</b>	configure the window watchdog timer counter value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>counter_value</b>	counter_value: 0x0000 - 0x007F
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* update WWDGT counter to 0x7F */

wwdgt_counter_update(127);
```

### wwdgt\_config

The description of wwdgt\_config is shown as below:

**Table 3-1431. Function wwdgt\_config**

<b>Function name</b>	wwdgt_config
<b>Function prototype</b>	void wwdgt_config(uint16_t counter, uint16_t window, uint32_t prescaler);
<b>Function descriptions</b>	configure counter value, window value, and prescaler divider value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>counter</b>	counter: 0x0000 - 0x007F
<b>Input parameter{in}</b>	
<b>window</b>	window: 0x0000 - 0x007F
<b>Input parameter{in}</b>	
<b>prescaler</b>	wwdgt prescaler value
<i>WWDGT_CFG_PSC_D IV1</i>	the time base of WWDGT counter = (PCLK3/4096)/1

WWDGT_CFG_PSC_D IV2	the time base of WWDGT counter = (PCLK3/4096)/2
WWDGT_CFG_PSC_D IV4	the time base of WWDGT counter = (PCLK3/4096)/4
WWDGT_CFG_PSC_D IV8	the time base of WWDGT counter = (PCLK3/4096)/8
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 3-1432. 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 3-1433. Function wwdgt\_flag\_get**

Function name	wwdgt_flag_get
Function prototype	FlagStatus wwdgt_flag_get(void);

<b>Function descriptions</b>	check early wakeup interrupt state of WWDGT
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-1434. Function wwdgt\_flag\_clear**

<b>Function name</b>	wwdgt_flag_clear
<b>Function prototype</b>	void wwdgt_flag_clear(void);
<b>Function descriptions</b>	clear early wakeup interrupt state of WWDGT
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear early wakeup interrupt state of WWDGT */
```

```
wwdgt_flag_clear();
```



## 4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Nov.8, 2024
1.1	<p>1. Add LPTIMER_EXTRIGGER_RTCTAMP1 and LPTIMER_EXTRIGGER_RTCTAMP2 in <b><u>Table 3-711</u></b>.</p> <p>2. Add TRIGSEL_INPUT_RTC_TAMP1 and TRIGSEL_INPUT_RTC_TAMP2 in <b><u>Table 3-1285</u></b>.</p> <p>3. Change hrtimer_comparecfg_struct_para_init(comparecfg_para) to hrtimer_comparecfg_struct_para_init(&amp;comparecfg_para) in <b><u>chapter 3.27.2</u></b>.</p> <p>4. Modify the description of timersrc for function hrtimer_output_exchange in <b><u>Table 3-963</u></b>.</p> <p>5. Add called function master_timer_waveform_config/slave_timer_waveform_config for hrtimer_timers_waveform_init in <b><u>Table 3-965</u></b> and channel_output_config in <b><u>Table 3-971</u></b>.</p> <p>6. Delete the input parameter timer_id of hrtimer_slavetimer_waveform_compare_config in <b><u>Table 3-969</u></b>, function hrtimer_slavetimer_waveform_channel_config in <b><u>Table 3-971</u></b>, function hrtimer_slavetimer_waveform_channel_software_request in <b><u>Table 3-972</u></b>, function hrtimer_slavetimer_waveform_channel_output_level_get in <b><u>Table 3-973</u></b>, function hrtimer_slavetimer_waveform_channel_state_get in <b><u>Table 3-974</u></b>.</p> <p>7. Change hrtimer_slavetimer_capture_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t capturex, uint32_t trgsource) to hrtimer_slavetimer_capture_config(uint32_t hrtimer_periph, uint32_t timer_id, uint32_t capturex, uint64_t trgsource) in <b><u>Table 3-1025</u></b>.</p>	Feb.11, 2025

Revision No.	Description	Date
	<p>8. Update <b><u>Figure 2-2</u></b> and <b><u>Figure 2-5</u></b>.</p> <p>9. Change TIMER4_CIO_INPUT_IRC40K /  TIMER15_CIO_INPUT_IRC40K /  TIMER16_CIO_INPUT_IRC40K to  TIMER4_CIO_INPUT_IRC32K /  TIMER15_CIO_INPUT_IRC32K /  TIMER16_CIO_INPUT_IRC32K in <b><u>Table 3-1102</u></b>.</p> <p>10. Add function mpu_enable / mpu_disable /  mpu_attribute_struct_para_init /  mpu_attribute_config.</p>	

## 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.