

**GigaDevice Semiconductor Inc.**

**GD32M53x**

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

**Firmware Library  
User Guide**

Revision 1.0

(Feb. 2026)

# Table of Contents

<b>Table of Contents .....</b>	<b>1</b>
<b>List of Figures .....</b>	<b>4</b>
<b>List of Tables .....</b>	<b>5</b>
<b>1. Introduction .....</b>	<b>29</b>
<b>1.1. Rules of User Manual and Firmware Library .....</b>	<b>29</b>
1.1.1. Peripherals.....	29
1.1.2. Naming rules.....	30
<b>2. Firmware Library Overview.....</b>	<b>32</b>
<b>2.1. File Structure of Firmware Library .....</b>	<b>32</b>
2.1.1. Examples Folder .....	33
2.1.2. Firmware Folder.....	33
2.1.3. Template Folder .....	33
2.1.4. Utilities Folder .....	36
<b>2.2. File descriptions of Firmware Library .....</b>	<b>36</b>
<b>3. Firmware Library of Standard Peripherals .....</b>	<b>37</b>
<b>3.1. Overview of Firmware Library of Standard Peripherals.....</b>	<b>37</b>
<b>3.2. ADC .....</b>	<b>37</b>
3.2.1. Descriptions of Peripheral registers.....	37
3.2.2. Descriptions of Peripheral functions .....	39
<b>3.3. CAN .....</b>	<b>96</b>
3.3.1. Descriptions of Peripheral registers.....	97
3.3.2. Descriptions of Peripheral functions .....	97
<b>3.4. CFMU .....</b>	<b>114</b>
3.4.1. Descriptions of Peripheral registers.....	114
3.4.2. Descriptions of Peripheral functions .....	114
<b>3.5. CMP .....</b>	<b>124</b>
3.5.1. Descriptions of Peripheral registers.....	124
3.5.2. Descriptions of Peripheral functions .....	124
<b>3.6. CPTIMER .....</b>	<b>137</b>
3.6.1. Descriptions of Peripheral registers.....	137
3.6.2. Descriptions of Peripheral functions .....	137
<b>3.7. CPTIMERW.....</b>	<b>147</b>
3.7.1. Descriptions of Peripheral registers.....	147
3.7.2. Descriptions of Peripheral functions .....	148

<b>3.8. CRC .....</b>	<b>164</b>
3.8.1. Descriptions of Peripheral registers .....	164
3.8.2. Descriptions of Peripheral functions .....	164
<b>3.9. DAC .....</b>	<b>173</b>
3.9.1. Descriptions of Peripheral registers .....	173
3.9.2. Descriptions of Peripheral functions .....	173
<b>3.10. DBG .....</b>	<b>187</b>
3.10.1. Descriptions of Peripheral registers .....	187
3.10.2. Descriptions of Peripheral functions .....	187
<b>3.11. DMA&amp;DMAMUX .....</b>	<b>194</b>
3.11.1. Descriptions of Peripheral registers .....	194
3.11.2. Descriptions of Peripheral functions .....	195
<b>3.12. EVIC .....</b>	<b>242</b>
3.12.1. Descriptions of Peripheral registers .....	242
3.12.2. Descriptions of Peripheral functions .....	243
<b>3.13. EXTI.....</b>	<b>262</b>
3.13.1. Descriptions of Peripheral registers .....	262
3.13.2. Descriptions of Peripheral functions .....	262
<b>3.14. FMC .....</b>	<b>271</b>
3.14.1. Descriptions of Peripheral registers .....	271
3.14.2. Descriptions of Peripheral functions .....	272
<b>3.15. FWDGT.....</b>	<b>303</b>
3.15.1. Descriptions of Peripheral registers .....	303
3.15.2. Descriptions of Peripheral functions .....	303
<b>3.16. GPIO .....</b>	<b>309</b>
3.16.1. Descriptions of Peripheral registers .....	309
3.16.2. Descriptions of Peripheral functions .....	310
<b>3.17. GPTIMER.....</b>	<b>324</b>
3.17.1. Descriptions of Peripheral registers .....	324
3.17.2. Descriptions of Peripheral functions .....	325
<b>3.18. GTOC .....</b>	<b>406</b>
3.18.1. Descriptions of Peripheral registers .....	406
3.18.2. Descriptions of Peripheral functions .....	406
<b>3.19. I2C .....</b>	<b>421</b>
3.19.1. Descriptions of Peripheral registers .....	421
3.19.2. Descriptions of Peripheral functions .....	421
<b>3.20. MISC.....</b>	<b>457</b>
3.20.1. Descriptions of Peripheral registers .....	457
3.20.2. Descriptions of Peripheral functions .....	457

<b>3.21. PMU.....</b>	<b>464</b>
3.21.1. Descriptions of Peripheral registers.....	464
3.21.2. Descriptions of Peripheral functions.....	465
<b>3.22. POC.....</b>	<b>478</b>
3.22.1. Descriptions of Peripheral registers.....	478
3.22.2. Descriptions of Peripheral functions.....	479
<b>3.23. RCU.....</b>	<b>505</b>
3.23.1. Descriptions of Peripheral registers.....	506
3.23.2. Descriptions of Peripheral functions.....	506
<b>3.24. SPI.....</b>	<b>535</b>
3.24.1. Descriptions of Peripheral registers.....	535
3.24.2. Descriptions of Peripheral functions.....	536
<b>3.25. SVPWM.....</b>	<b>555</b>
3.25.1. Descriptions of Peripheral registers.....	555
3.25.2. Descriptions of Peripheral functions.....	556
<b>3.26. SYSCFG.....</b>	<b>560</b>
3.26.1. Descriptions of Peripheral registers.....	560
3.26.2. Descriptions of Peripheral functions.....	561
<b>3.27. TIMER.....</b>	<b>579</b>
3.27.1. Descriptions of Peripheral registers.....	579
3.27.2. Descriptions of Peripheral functions.....	580
<b>3.28. TMU.....</b>	<b>673</b>
3.28.1. Descriptions of Peripheral registers.....	673
3.28.2. Descriptions of Peripheral functions.....	673
<b>3.29. UART.....</b>	<b>687</b>
3.29.1. Descriptions of Peripheral registers.....	687
3.29.2. Descriptions of Peripheral functions.....	687
<b>3.30. WWDGT.....</b>	<b>711</b>
3.30.1. Descriptions of Peripheral registers.....	711
3.30.2. Descriptions of Peripheral functions.....	712
<b>4. Revision history.....</b>	<b>717</b>



## List of Figures

Figure 2-1. File structure of firmware library of GD32M53x.....	32
Figure 2-2. Select peripheral example files .....	34
Figure 2-3. Copy the peripheral example files .....	34
Figure 2-4. Open the project file .....	35
Figure 2-5. Configure project files .....	35
Figure 2-6. Compile-debug-download .....	35

# List of Tables

Table 1-1. Peripherals .....	29
Table 2-1. Function descriptions of Firmware Library .....	36
Table 3-1. Peripheral function format of Firmware Library .....	37
Table 3-2. ADC Registers .....	37
Table 3-3. ADC firmware function.....	39
Table 3-4. Enum adc_group_select_enum.....	41
Table 3-5. Enum adc_channel_select_enum .....	41
Table 3-6. Enum adc_bifurcate_data_enum .....	42
Table 3-7. Enum adc_disc_detect_mode_enum .....	42
Table 3-8. Enum adc_self_diag_mode_enum.....	42
Table 3-9. Enum adc_self_diag_fixed_voltage_enum .....	42
Table 3-10. Enum adc_restart_channel_enum .....	42
Table 3-11. Enum adc_watchdog_select_enum.....	43
Table 3-12. Enum adc_watchdog_compare_condition_enum .....	43
Table 3-13. Enum adc_oversample_mode_enum.....	43
Table 3-14. Enum adc_interrupt_enum.....	43
Table 3-15. Enum adc_interrupt_flag_enum.....	44
Table 3-16. Enum adc_event_flag_enum .....	44
Table 3-17. Enum adc_evic_link_source_enum.....	45
Table 3-18. Function adc_deinit.....	45
Table 3-19. Function adc_enable .....	46
Table 3-20. Function adc_disable .....	46
Table 3-21. Function adc_data_alignment_config.....	47
Table 3-22. Function adc_resolution_config .....	47
Table 3-23. Function adc_self_diagnosis_enable .....	48
Table 3-24. Function adc_self_diagnosis_disable .....	49
Table 3-25. Function adc_self_diagnosis_mode_config .....	49
Table 3-26. Function adc_disconnect_detect_mode_config .....	50
Table 3-27. Function adc_disconnect_detect_period_config .....	50
Table 3-28. Function adc_data_auto_clear_enable .....	51
Table 3-29. Function adc_data_auto_clear_disable .....	51
Table 3-30. Function adc_data_auto_set_enable .....	52
Table 3-31. Function adc_data_auto_set_disable .....	52
Table 3-32. Function adc_sample_hold_channel_config.....	53
Table 3-33. Function adc_sh_sample_time_config .....	54
Table 3-34. Function adc_sh_hold_time_config .....	54
Table 3-35. Function adc_sh_constant_sampling_mode_enable .....	55
Table 3-36. Function adc_sh_constant_sampling_mode_disable .....	55
Table 3-37. Function adc_sh_constant_sampling_start .....	56
Table 3-38. Function adc_sh_constant_sampling_stop.....	56
Table 3-39. Function adc_watchdog_enable .....	56

Table 3-40. Function <code>adc_watchdog_disable</code> .....	57
Table 3-41. Function <code>adc_watchdog_a_channel_config</code> .....	58
Table 3-42. Function <code>adc_watchdog_a_channel_deselect</code> .....	58
Table 3-43. Function <code>adc_watchdog_b_channel_config</code> .....	59
Table 3-44. Function <code>adc_watchdog_a_threshold_config</code> .....	60
Table 3-45. Function <code>adc_watchdog_b_threshold_config</code> .....	60
Table 3-46. Function <code>adc_watchdog_window_mode_enable</code> .....	61
Table 3-47. Function <code>adc_watchdog_window_mode_disable</code> .....	61
Table 3-48. Function <code>adc_watchdog_complex_condition_config</code> .....	62
Table 3-49. Function <code>adc_oversample_channel_config</code> .....	63
Table 3-50. Function <code>adc_oversample_mode_config</code> .....	64
Table 3-51. Function <code>adc_oversample_mode_enable</code> .....	64
Table 3-52. Function <code>adc_oversample_mode_disable</code> .....	65
Table 3-53. Function <code>adc_group_scan_mode_config</code> .....	65
Table 3-54. Function <code>adc_group_priority_control_enable</code> .....	66
Table 3-55. Function <code>adc_group_priority_control_disable</code> .....	66
Table 3-56. Function <code>adc_group_pri3_enable</code> .....	67
Table 3-57. Function <code>adc_group_pri3_disable</code> .....	67
Table 3-58. Function <code>adc_group_pri4_enable</code> .....	68
Table 3-59. Function <code>adc_group_pri4_disable</code> .....	68
Table 3-60. Function <code>adc_group_lowest_priority_continuous_enable</code> .....	69
Table 3-61. Function <code>adc_group_lowest_priority_continuous_disable</code> .....	69
Table 3-62. Function <code>adc_group_restart_enable</code> .....	70
Table 3-63. Function <code>adc_group_restart_disable</code> .....	71
Table 3-64. Function <code>adc_group_channel_config</code> .....	71
Table 3-65. Function <code>adc_group_channel_deselect</code> .....	72
Table 3-66. Function <code>adc_group_end_flag_round_config</code> .....	73
Table 3-67. Function <code>adc_group_external_trigger_enable</code> .....	73
Table 3-68. Function <code>adc_group_extern_trigger_edge_config</code> .....	74
Table 3-69. Function <code>adc_group_external_trigger_disable</code> .....	75
Table 3-70. Function <code>adc_group_synchronous_trigger_enable</code> .....	75
Table 3-71. Function <code>adc_group_synchronous_trigger_source_config</code> .....	76
Table 3-72. Function <code>adc_group_asynchronous_trigger_enable</code> .....	78
Table 3-73. Function <code>adc_group_software_trigger_enable</code> .....	79
Table 3-74. Function <code>adc_group_software_end_conversion</code> .....	79
Table 3-75. Function <code>adc_group_bifurcate_mode_enable</code> .....	80
Table 3-76. Function <code>adc_group_bifurcate_mode_disable</code> .....	80
Table 3-77. Function <code>adc_group_bifurcate_channel_select</code> .....	81
Table 3-78. Function <code>adc_group_bifurcate_extend_trigger_select</code> .....	82
Table 3-79. Function <code>adc_group_bifurcate_trigger_restart_enable</code> .....	83
Table 3-80. Function <code>adc_group_bifurcate_trigger_restart_disable</code> .....	83
Table 3-81. Function <code>adc_group_dma_mode_enable</code> .....	84
Table 3-82. Function <code>adc_group_dma_mode_disable</code> .....	84
Table 3-83. Function <code>adc_group_dma_request_after_last_enable</code> .....	85

Table 3-84. Function <code>adc_group_dma_request_after_last_disable</code> .....	86
Table 3-85. Function <code>adc_group_dma_overrun_detect_enable</code> .....	86
Table 3-86. Function <code>adc_group_dma_overrun_detect_disable</code> .....	87
Table 3-87. Function <code>adc_channel_priority_config</code> .....	87
Table 3-88. Function <code>adc_channel_sample_time_config</code> .....	88
Table 3-89. Function <code>adc_evic_link_signal_event_config</code> .....	89
Table 3-90. Function <code>adc_group_evic_link_signal_source</code> .....	90
Table 3-91. Function <code>adc_channel_data_read</code> .....	90
Table 3-92. Function <code>adc_self_diagnosis_data_read</code> .....	91
Table 3-93. Function <code>adc_self_diagnosis_status_read</code> .....	92
Table 3-94. Function <code>adc_bifurcate_data_read</code> .....	92
Table 3-95. Function <code>adc_flag_get</code> .....	93
Table 3-96. Function <code>adc_flag_clear</code> .....	94
Table 3-97. Function <code>adc_interrupt_enable</code> .....	94
Table 3-98. Function <code>adc_interrupt_disable</code> .....	95
Table 3-99. Function <code>adc_interrupt_flag_get</code> .....	95
Table 3-100. Function <code>adc_interrupt_flag_clear</code> .....	96
Table 3-101. CAN Registers .....	97
Table 3-102. CAN firmware function .....	97
Table 3-103. Structure <code>can_parameter_struct</code> .....	98
Table 3-104. Structure <code>can_transmit_message_struct</code> .....	99
Table 3-105. Structure <code>can_receive_message_struct</code> .....	99
Table 3-106. Structure <code>can_filter_parameter_struct</code> .....	99
Table 3-107. Function <code>can_deinit</code> .....	99
Table 3-108. Function <code>can_struct_para_init</code> .....	100
Table 3-109. Function <code>can_init</code> .....	101
Table 3-110. Function <code>can_filter_init</code> .....	101
Table 3-111. Function <code>can_debug_freeze_enable</code> .....	102
Table 3-112. Function <code>can_debug_freeze_disable</code> .....	102
Table 3-113. Function <code>can_time_trigger_mode_enable</code> .....	103
Table 3-114. Function <code>can_time_trigger_mode_disable</code> .....	103
Table 3-115. Function <code>can_message_transmit</code> .....	103
Table 3-116. Function <code>can_transmit_states</code> .....	104
Table 3-117. Function <code>can_transmission_stop</code> .....	105
Table 3-118. Function <code>can_message_receive</code> .....	105
Table 3-119. Function <code>can_fifo_release</code> .....	106
Table 3-120. Function <code>can_receive_message_length_get</code> .....	106
Table 3-121. Function <code>can_working_mode_set</code> .....	107
Table 3-122. Function <code>can_wakeup</code> .....	107
Table 3-123. Function <code>can_error_get</code> .....	108
Table 3-124. Function <code>can_receive_error_number_get</code> .....	108
Table 3-125. Function <code>can_transmit_error_number_get</code> .....	109
Table 3-126. Function <code>can_flag_get</code> .....	109
Table 3-127. Function <code>can_flag_clear</code> .....	110

Table 3-128. Function can_interrupt_enable .....	111
Table 3-129. Function can_interrupt_disable .....	112
Table 3-130. Function can_interrupt_flag_get.....	112
Table 3-131. Function can_interrupt_flag_clear .....	113
Table 3-132. CFMU Registers .....	114
Table 3-133. CFMU firmware function.....	114
Table 3-134. Enum cfmu_int_enum .....	115
Table 3-135. Enum cfmu_int_flag_enum .....	115
Table 3-136. Enum cfmu_int_flag_clear_enum .....	115
Table 3-137. Function cfmu_deinit .....	116
Table 3-138. Function cfmu_enable .....	116
Table 3-139. Function cfmu_disable .....	117
Table 3-140. Function cfmu_cfmuref_enable .....	117
Table 3-141. Function cfmu_cfmuref_disable .....	118
Table 3-142. Function cfmu_reference_signal_config.....	118
Table 3-143. Function cfmu_digital_filter_config .....	119
Table 3-144. Function cfmu_reference_clock_config .....	119
Table 3-145. Function cfmu_measurement_clock_config .....	120
Table 3-146. Function cfmu_limit_value_config .....	121
Table 3-147. Function cfmu_interrupt_enable.....	121
Table 3-148. Function cfmu_interrupt_disable.....	122
Table 3-149. Function cfmu_interrupt_flag_get .....	123
Table 3-150. Function cfmu_interrupt_flag_clear .....	123
Table 3-151. CMP registers .....	124
Table 3-152. CMP firmware function .....	124
Table 3-153. Enum cmp_enum .....	125
Table 3-154. Function cmp_deinit .....	125
Table 3-155. Function cmp_mode_init.....	126
Table 3-156. Function cmp_noninverting_input_select .....	127
Table 3-157. Function cmp_output_init .....	128
Table 3-158. Function cmp_digital_filter_init .....	128
Table 3-159. Function cmp_enable.....	129
Table 3-160. Function cmp_disable .....	130
Table 3-161. Function cmp_lock_enable .....	130
Table 3-162. Function cmp_output_to_pin_enable .....	131
Table 3-163. Function cmp_output_to_pin_disable .....	131
Table 3-164. Function cmp_output_enable.....	132
Table 3-165. Function cmp_output_disable.....	132
Table 3-166. Function cmp_output_level_get .....	133
Table 3-167. Function cmp_flag_get .....	133
Table 3-168. Function cmp_flag_clear .....	134
Table 3-169. Function cmp_interrupt_enable.....	134
Table 3-170. Function cmp_interrupt_disable.....	135
Table 3-171. Function cmp_interrupt_flag_get .....	136

Table 3-172. Function <code>cmp_interrupt_flag_clear</code> .....	136
Table 3-173. CPTIMER registers .....	137
Table 3-174. CPTIMER firmware function.....	137
Table 3-175. Function <code>cptimer_deinit</code> .....	138
Table 3-176. Function <code>cptimer_enable</code> .....	138
Table 3-177. Function <code>cptimer_disable</code> .....	139
Table 3-178. Function <code>cptimer_prescaler_config</code> .....	139
Table 3-179. Function <code>cptimer_autoreload_value_config</code> .....	140
Table 3-180. Function <code>cptimer_counter_value_config</code> .....	141
Table 3-181. Function <code>cptimer_counter_read</code> .....	141
Table 3-182. Function <code>cptimer_prescaler_read</code> .....	142
Table 3-183. Function <code>cptimer_event_software_generate</code> .....	143
Table 3-184. Function <code>cptimer_flag_get</code> .....	143
Table 3-185. Function <code>cptimer_flag_clear</code> .....	144
Table 3-186. Function <code>cptimer_interrupt_enable</code> .....	145
Table 3-187. Function <code>cptimer_interrupt_disable</code> .....	145
Table 3-188. Function <code>cptimer_interrupt_flag_get</code> .....	146
Table 3-189. Function <code>cptimer_interrupt_flag_clear</code> .....	147
Table 3-190. CPTIMERW registers .....	147
Table 3-191. CPTIMERW firmware function.....	148
Table 3-192. <code>cptimerw_init_parameter_struct</code> .....	149
Table 3-193. <code>cptimerw_ic_parameter_struct</code> .....	149
Table 3-194. Function <code>cptimerw_deinit</code> .....	149
Table 3-195. Function <code>cptimerw_struct_para_init</code> .....	150
Table 3-196. Function <code>cptimerw_init</code> .....	150
Table 3-197. Function <code>cptimerw_autoreload_value_config</code> .....	151
Table 3-198. Function <code>cptimerw_lock_enable</code> .....	151
Table 3-199. Function <code>cptimerw_counter_value_config</code> .....	152
Table 3-200. Function <code>cptimerw_counter_clear_source_config</code> .....	152
Table 3-201. Function <code>cptimerw_counter_read</code> .....	153
Table 3-202. Function <code>cptimerw_output_disable</code> .....	153
Table 3-203. Function <code>cptimerw_enable</code> .....	154
Table 3-204. Function <code>cptimerw_disable</code> .....	154
Table 3-205. Function <code>cptimerw_channel_output_mode_select</code> .....	155
Table 3-206. Function <code>cptimerw_channel_output_compare_value_config</code> .....	156
Table 3-207. Function <code>cptimerw_channel_output_state_config</code> .....	156
Table 3-208. Function <code>cptimerw_channel_input_struct_para_init</code> .....	157
Table 3-209. Function <code>cptimerw_input_capture_config</code> .....	157
Table 3-210. Function <code>cptimerw_channel_capture_value_register_read</code> .....	158
Table 3-211. Function <code>cptimerw_event_software_generate</code> .....	159
Table 3-212. Function <code>cptimerw_flag_get</code> .....	159
Table 3-213. Function <code>cptimerw_flag_clear</code> .....	160
Table 3-214. Function <code>cptimerw_interrupt_enable</code> .....	161
Table 3-215. Function <code>cptimerw_interrupt_disable</code> .....	161

Table 3-216. Function <code>cptimerw_interrupt_flag_get</code> .....	162
Table 3-217. Function <code>cptimerw_interrupt_flag_clear</code> .....	163
Table 3-218. CRC Registers .....	164
Table 3-219. CRC firmware function .....	164
Table 3-220. Function <code>crc_deinit</code> .....	165
Table 3-221. Function <code>crc_data_register_reset</code> .....	165
Table 3-222. Function <code>crc_reverse_output_data_enable</code> .....	166
Table 3-223. Function <code>crc_reverse_output_data_disable</code> .....	166
Table 3-224. Function <code>crc_input_data_reverse_config</code> .....	166
Table 3-225. Function <code>crc_data_register_read</code> .....	167
Table 3-226. Function <code>crc_free_data_register_read</code> .....	168
Table 3-227. Function <code>crc_free_data_register_write</code> .....	168
Table 3-228. Function <code>crc_init_data_register_write</code> .....	169
Table 3-229. Function <code>crc_polynomial_size_set</code> .....	169
Table 3-230. Function <code>crc_polynomial_set</code> .....	170
Table 3-231. Function <code>crc_single_data_calculate</code> .....	170
Table 3-232. Function <code>crc_block_data_calculate</code> .....	171
Table 3-233. DAC Registers .....	173
Table 3-234. DAC firmware functions .....	173
Table 3-235. Function <code>dac_deinit</code> .....	174
Table 3-236. Function <code>dac_enable</code> .....	174
Table 3-237. Function <code>dac_disable</code> .....	175
Table 3-238. Function <code>dac_sync_enable</code> .....	175
Table 3-239. Function <code>dac_sync_disable</code> .....	176
Table 3-240. Function <code>dac_connect_to_pin_enable</code> .....	177
Table 3-241. Function <code>dac_connect_to_pin_disable</code> .....	177
Table 3-242. Function <code>dac_connect_to_cmp_enable</code> .....	178
Table 3-243. Function <code>dac_connect_to_cmp_disable</code> .....	178
Table 3-244. Function <code>dac_output_value_get</code> .....	179
Table 3-245. Function <code>dac_data_set</code> .....	180
Table 3-246. Function <code>dac_trigger_enable</code> .....	180
Table 3-247. Function <code>dac_trigger_disable</code> .....	181
Table 3-248. Function <code>dac_trigger_source_config</code> .....	181
Table 3-249. Function <code>dac_software_trigger_enable</code> .....	182
Table 3-250. Function <code>dac_wave_mode_config</code> .....	183
Table 3-251. Function <code>dac_lfsr_noise_config</code> .....	183
Table 3-252. Function <code>dac_triangle_noise_config</code> .....	184
Table 3-253. Function <code>dac_concurrent_enable</code> .....	185
Table 3-254. Function <code>dac_concurrent_disable</code> .....	185
Table 3-255. Function <code>dac_concurrent_software_trigger_enable</code> .....	186
Table 3-256. Function <code>dac_concurrent_data_set</code> .....	186
Table 3-257. DBG Registers .....	187
Table 3-258. DBG firmware function .....	187
Table 3-259. Enum <code>dbg_periph_enum</code> .....	188

Table 3-260. Function dbg_deinit .....	188
Table 3-261. Function dbg_id_get .....	189
Table 3-262. Function dbg_low_power_enable.....	190
Table 3-263. Function dbg_low_power_disable.....	191
Table 3-264. Function dbg_periph_enable .....	191
Table 3-265. Function dbg_periph_disable .....	192
Table 3-266. Function dbg_trace_pin_enable .....	193
Table 3-267. Function dbg_trace_pin_disable .....	193
Table 3-268. DMA Registers.....	194
Table 3-269. DMAMUX Registers .....	194
Table 3-270. DMA firmware function .....	195
Table 3-271. DMAMUX firmware function.....	196
Table 3-272. Structure dma_parameter_struct.....	196
Table 3-273. Structure dmamux_sync_parameter_struct .....	197
Table 3-274. Structure dmamux_gen_parameter_struct .....	197
Table 3-275. Enum dmamux_interrupt_enum .....	197
Table 3-276. Enum dmamux_flag_enum .....	198
Table 3-277. Enum dmamux_interrupt_flag_enum.....	199
Table 3-278. Enum dma_channel_enum .....	200
Table 3-279. Enum dmamux_multiplexer_channel_enum .....	200
Table 3-280. Enum dmamux_generator_channel_enum .....	200
Table 3-281. Function dma_deinit .....	201
Table 3-282. Function dma_para_init.....	201
Table 3-283. Function dma_init.....	202
Table 3-284. Function dma_circulation_enable .....	203
Table 3-285. Function dma_circulation_disable .....	203
Table 3-286. Function dma_memory_to_memory_enable .....	204
Table 3-287. Function dma_memory_to_memory_disable .....	204
Table 3-288. Function dma_channel_enable .....	205
Table 3-289. Function dma_channel_disable .....	206
Table 3-290. Function dma_periph_address_config .....	206
Table 3-291. Function dma_memory_address_config.....	207
Table 3-292. Function dma_transfer_number_config .....	207
Table 3-293. Function dma_transfer_number_get.....	208
Table 3-294. Function dma_priority_config .....	209
Table 3-295. Function dma_memory_width_config .....	209
Table 3-296. Function dma_periph_width_config .....	210
Table 3-297. Function dma_memory_increase_enable .....	211
Table 3-298. Function dma_memory_increase_disable .....	212
Table 3-299. Function dma_periph_increase_enable .....	212
Table 3-300. Function dma_periph_increase_disable .....	213
Table 3-301. Function dma_transfer_direction_config.....	213
Table 3-302. Function dma_flag_get .....	214
Table 3-303. Function dma_flag_clear .....	215



Table 3-304. Function dma_interrupt_enable.....	216
Table 3-305. Function dma_interrupt_disable.....	216
Table 3-306. Function dma_interrupt_flag_get .....	217
Table 3-307. Function dma_interrupt_flag_clear .....	218
Table 3-308. Function dmamux_sync_struct_para_init.....	218
Table 3-309. Function dmamux_synchronization_init .....	219
Table 3-310. Function dmamux_synchronization_enable.....	220
Table 3-311. Function dmamux_synchronization_disable.....	220
Table 3-312. Function dmamux_event_generation_enable .....	221
Table 3-313. Function dmamux_event_generation_disable .....	221
Table 3-314. Function dmamux_gen_struct_para_init .....	222
Table 3-315. Function dmamux_request_generator_init.....	223
Table 3-316. Function dmamux_request_generator_channel_enable .....	223
Table 3-317. Function dmamux_request_generator_channel_disable.....	224
Table 3-318. Function dmamux_synchronization_polarity_config .....	224
Table 3-319. Function dmamux_request_forward_number_config.....	225
Table 3-320. Function dmamux_sync_id_config .....	226
Table 3-321. Function dmamux_request_id_config .....	227
Table 3-322. Function dmamux_trigger_polarity_config .....	232
Table 3-323. Function dmamux_request_generate_number_config.....	233
Table 3-324. Function dmamux_trigger_id_config.....	233
Table 3-325. Function dmamux_flag_get .....	235
Table 3-326. Function dmamux_flag_clear .....	236
Table 3-327. Function dmamux_interrupt_enable .....	237
Table 3-328. Function dmamux_interrupt_disable .....	238
Table 3-329. Function dmamux_interrupt_flag_get .....	240
Table 3-330. Function dmamux_interrupt_flag_clear .....	241
Table 3-331. EVIC registers.....	242
Table 3-332. EVIC firmware function.....	243
Table 3-333. Enum event_source_enum .....	244
Table 3-334. Enum evic_periph_enum.....	247
Table 3-335. Enum evic_cptimer_enum.....	249
Table 3-336. Enum evic_single_io_enum.....	249
Table 3-337. Function evic_init .....	249
Table 3-338. Function evic_event_source_get.....	250
Table 3-339. Function evic_register_lock_set.....	250
Table 3-340. Function evic_register_lock_get.....	251
Table 3-341. Function evic_cptimer_slave_mode_select.....	251
Table 3-342. Function evic_group_member_config.....	252
Table 3-343. Function evic_group_edge_detection_config .....	253
Table 3-344. Function evic_group_output_level_config .....	254
Table 3-345. Function evic_group_overwrite_enable .....	254
Table 3-346. Function evic_group_overwrite_disable .....	255
Table 3-347. Function evic_data_set.....	255

Table 3-348. Function <code>evic_data_get</code> .....	256
Table 3-349. Function <code>evic_single_io_config</code> .....	257
Table 3-350. Function <code>evic_single_io_config</code> .....	257
Table 3-351. Function <code>evic_single_io_output_level_config</code> .....	258
Table 3-352. Function <code>evic_register_write_enable</code> .....	259
Table 3-353. Function <code>evic_register_write_disable</code> .....	259
Table 3-354. Function <code>evic_register_write_enable_get</code> .....	260
Table 3-355. Function <code>evic_bit_write_enable</code> .....	260
Table 3-356. Function <code>evic_bit_write_disable</code> .....	261
Table 3-357. Function <code>evic_register_write_enable_get</code> .....	261
Table 3-358. Function <code>evic_software_event_generation</code> .....	262
Table 3-359. EXTI Registers .....	262
Table 3-360. EXTI firmware function .....	263
Table 3-361. <code>exti_line_enum</code> .....	263
Table 3-362. <code>exti_mode_enum</code> .....	264
Table 3-363. <code>exti_trig_type_enum</code> .....	264
Table 3-364. Function <code>exti_deinit</code> .....	264
Table 3-365. Function <code>exti_init</code> .....	265
Table 3-366. Function <code>exti_interrupt_enable</code> .....	265
Table 3-367. Function <code>exti_interrupt_disable</code> .....	266
Table 3-368. Function <code>exti_event_enable</code> .....	266
Table 3-369. Function <code>exti_event_disable</code> .....	267
Table 3-370. Function <code>exti_software_interrupt_enable</code> .....	267
Table 3-371. Function <code>exti_software_interrupt_disable</code> .....	267
Table 3-372. Function <code>exti_digital_filter_enable</code> .....	268
Table 3-373. Function <code>exti_digital_filter_disable</code> .....	269
Table 3-374. Function <code>exti_flag_get</code> .....	269
Table 3-375. Function <code>exti_flag_clear</code> .....	270
Table 3-376. Function <code>exti_interrupt_flag_get</code> .....	270
Table 3-377. Function <code>exti_interrupt_flag_clear</code> .....	271
Table 3-378. FMC Registers .....	271
Table 3-379. FMC firmware function .....	272
Table 3-380. Structure <code>optionbyte_user_struct</code> .....	273
Table 3-381. Structure <code>optionbyte_wwdgt0_struct</code> .....	273
Table 3-382. <code>fmc_state_enum</code> .....	274
Table 3-383. <code>fmc_flag_enum</code> .....	274
Table 3-384. <code>fmc_interrupt_flag_enum</code> .....	275
Table 3-385. <code>fmc_interrupt_enum</code> .....	275
Table 3-386. <code>fmc_ob_wp_enum</code> .....	276
Table 3-387. Function <code>fmc_unlock</code> .....	277
Table 3-388. Function <code>fmc_lock</code> .....	277
Table 3-389. Function <code>fmc_wsnt_set</code> .....	277
Table 3-390. Function <code>fmc_prefetch_enable</code> .....	278
Table 3-391. Function <code>fmc_prefetch_disable</code> .....	279

Table 3-392. Function fmc_ibus_enable.....	279
Table 3-393. Function fmc_ibus_disable.....	280
Table 3-394. Function fmc_ibus_reset_enable .....	280
Table 3-395. Function fmc_ibus_reset_disable .....	280
Table 3-396. Function fmc_dbus_enable .....	281
Table 3-397. Function fmc_dbus_disable .....	281
Table 3-398. Function fmc_dbus_reset_enable .....	282
Table 3-399. Function fmc_dbus_reset_disable .....	282
Table 3-400. Function fmc_blank_check.....	283
Table 3-401. Function fmc_page_erase.....	284
Table 3-402. Function fmc_mflash_mass_erase.....	284
Table 3-403. Function fmc_dflash_mass_erase.....	285
Table 3-404. Function fmc_mass_erase.....	286
Table 3-405. Function fmc_fourword_program.....	286
Table 3-406. Function fmc_fast_program .....	287
Table 3-407. Function fmc_otp_fourword_program.....	289
Table 3-408. Function ob_unlock .....	289
Table 3-409. Function ob_lock .....	290
Table 3-410. Function ob_erase.....	291
Table 3-411. Function ob_write_protection_enable .....	291
Table 3-412. Function ob_security_protection_config .....	292
Table 3-413. Function ob_user_write.....	292
Table 3-414. Function ob_data_program .....	293
Table 3-415. Function ob_user1_write.....	294
Table 3-416. Function ob_wwdgt0_write .....	295
Table 3-417. Function ob_wwdgt1_wwdgt2_write .....	296
Table 3-418. Function ob_fwdgt_write.....	296
Table 3-419. Function ob_user_get.....	297
Table 3-420. Function ob_data_program .....	298
Table 3-421. Function ob_write_protection_get .....	298
Table 3-422. Function ob_security_protection_flag_get.....	299
Table 3-423. Function fmc_ecc_error_address_get.....	299
Table 3-424. Function fmc_flag_get.....	300
Table 3-425. Function fmc_flag_clear .....	300
Table 3-426. Function fmc_interrupt_enable .....	301
Table 3-427. Function fmc_interrupt_disable.....	301
Table 3-428. Function fmc_interrupt_flag_get .....	302
Table 3-429. Function fmc_interrupt_flag_clear .....	302
Table 3-430. FWDGT Registers .....	303
Table 3-431. FWDGT firmware function.....	303
Table 3-432. Function fwdgt_write_ensable .....	304
Table 3-433. Function fwdgt_write_disable .....	304
Table 3-434. Function fwdgt_enable .....	305
Table 3-435. Function fwdgt_prescaler_value_config .....	305

Table 3-436. Function fwdgt_reload_value_config.....	306
Table 3-437. Function fwdgt_window_value_config .....	306
Table 3-438. Function fwdgt_counter_reload .....	307
Table 3-439. Function fwdgt_config .....	307
Table 3-440. Function fwdgt_reset_output .....	308
Table 3-441. Function fwdgt_interrupt_output.....	308
Table 3-442. Function fwdgt_flag_get.....	309
Table 3-443. Function gpio_port_write .....	315
Table 3-444. Function gpio_input_bit_get.....	315
Table 3-445. Function gpio_input_port_get.....	316
Table 3-446. Function gpio_output_bit_get .....	316
Table 3-447. Function gpio_output_port_get .....	317
Table 3-448. Function gpio_af_set .....	318
Table 3-449. Function gpio_pin_lock .....	319
Table 3-450. Function gpio_bit_toggle .....	319
Table 3-451. Function gpio_port_toggle .....	320
Table 3-452. GPTIMER registers .....	324
Table 3-453. GPTIMER firmware function .....	325
Table 3-454. Structure gptimer_parameter_struct.....	330
Table 3-455. Structure gptimer_counter_enable_source_parameter_struct .....	330
Table 3-456. Structure gptimer_counter_disable_source_parameter_struct.....	331
Table 3-457. Structure gptimer_counter_reset_source_parameter_struct.....	331
Table 3-458. Structure gptimer_capture_source_parameter_struct .....	332
Table 3-459. Structure gptimer_counter_up_source_parameter_struct .....	332
Table 3-460. Structure gptimer_counter_down_source_parameter_struct .....	333
Table 3-461. Structure gptimer_oc_parameter_struct .....	333
Table 3-462. Structure gptimer_com_oc_parameter_struct .....	334
Table 3-463. Function gptimer_deinit .....	334
Table 3-464. Function gptimer_struct_para_init .....	335
Table 3-465. Function gptimer_init.....	335
Table 3-466. Function gptimer_register_write_protect_enable .....	336
Table 3-467. Function gptimer_register_write_protect_disable .....	337
Table 3-468. Function gptimer_clock_source_select .....	338
Table 3-469. Function gptimer_clock_polarity_config .....	338
Table 3-470. Function gptimer_counter_software_enable.....	339
Table 3-471. Function gptimer_counter_software_disable.....	340
Table 3-472. Function gptimer_counter_software_reset.....	340
Table 3-473. Function gptimer_counter_enable_source_struct_para_init .....	341
Table 3-474. Function gptimer_counter_enable_source_config .....	341
Table 3-475. Function gptimer_counter_disable_source_struct_para_init .....	342
Table 3-476. Function gptimer_counter_disable_source_config .....	343
Table 3-477. Function gptimer_counter_reset_source_struct_para_init .....	343
Table 3-478. Function gptimer_counter_reset_source_config .....	344
Table 3-479. Function gptimer_counter_up_source_struct_para_init.....	345

Table 3-480. Function gptimer_counter_up_source_config .....	345
Table 3-481. Function gptimer_counter_down_source_struct_para_init .....	346
Table 3-482. Function gptimer_counter_down_source_config .....	346
Table 3-483. Function gptimer_counter_up_input_level_source_select .....	347
Table 3-484. Function gptimer_counter_down_input_level_source_select .....	348
Table 3-485. Function gptimer_enable .....	349
Table 3-486. Function gptimer_disable .....	350
Table 3-487. Function gptimer_auto_reload_shadow_enable .....	350
Table 3-488. Function gptimer_auto_reload_shadow_disable .....	351
Table 3-489. Function gptimer_update_event_enable .....	351
Table 3-490. Function gptimer_update_event_disable .....	352
Table 3-491. Function gptimer_counter_alignment .....	352
Table 3-492. Function gptimer_counter_up_direction .....	353
Table 3-493. Function gptimer_counter_down_direction .....	354
Table 3-494. Function gptimer_counter_direction_force_set_enable .....	354
Table 3-495. Function gptimer_counter_direction_force_set_disable .....	355
Table 3-496. Function gptimer_register_global_update_source_select .....	355
Table 3-497. Function gptimer_register_local_update_source_select .....	356
Table 3-498. Function gptimer_prescaler_config .....	357
Table 3-499. Function gptimer_update_repetition_value_config .....	358
Table 3-500. Function gptimer_update_repetition_counter_read .....	358
Table 3-501. Function gptimer_autoreload_value_config .....	359
Table 3-502. Function gptimer_counter_value_config .....	359
Table 3-503. Function gptimer_counter_read .....	360
Table 3-504. Function gptimer_prescaler_read .....	361
Table 3-505. Function gptimer_single_pulse_mode_config .....	361
Table 3-506. Function gptimer_dma_enable .....	362
Table 3-507. Function gptimer_dma_disable .....	363
Table 3-508. Function gptimer_dma_transfer_config .....	363
Table 3-509. Function gptimer_channel_output_struct_para_init .....	366
Table 3-510. Function gptimer_channel_output_config .....	366
Table 3-511. Function gptimer_channel_output_force_duty_config .....	367
Table 3-512. Function gptimer_channel_output_compare_value_config .....	368
Table 3-513. Function gptimer_channel_output_additional_compare_value_config .....	369
Table 3-514. Function gptimer_channel_output_shadow_config .....	370
Table 3-515. Function gptimer_channel_output_additional_shadow_config .....	370
Table 3-516. Function gptimer_channel_output_control_shadow_config .....	371
Table 3-517. Function gptimer_two_channel_output_high_check_enable .....	372
Table 3-518. Function gptimer_two_channel_output_high_check_disable .....	372
Table 3-519. Function gptimer_two_channel_output_low_check_enable .....	373
Table 3-520. Function gptimer_two_channel_output_low_check_disable .....	373
Table 3-521. Function gptimer_period_signal_output_enable .....	374
Table 3-522. Function gptimer_period_signal_output_disable .....	374
Table 3-523. Function gptimer_stop_output_set_select .....	375

Table 3-524. Function gptimer_stop_output_recover_time_select .....	376
Table 3-525. Function gptimer_channel_complementary_output_struct_para_init.....	376
Table 3-526. Function gptimer_channel_complementary_output_config.....	377
Table 3-527. Function gptimer_channel_io_direction_config .....	378
Table 3-528. Function gptimer_channel_io_state_config .....	378
Table 3-529. Function gptimer_capture_source_struct_para_init.....	379
Table 3-530. Function gptimer_capture_source_config.....	380
Table 3-531. Function gptimer_channel_input_capture_filter_config .....	380
Table 3-532. Function gptimer_channel_input_capture_prescaler_config .....	381
Table 3-533. Function gptimer_channel_capture_value_register_read .....	382
Table 3-534. Function gptimer_counter_sync_control_select .....	383
Table 3-535. Function gptimer_counter_sync_reset_source_select .....	383
Table 3-536. Function gptimer_trigger_adc_compare_enable.....	384
Table 3-537. Function gptimer_trigger_adc_compare_disable.....	385
Table 3-538. Function gptimer_trigger_adc_compare_value_config.....	386
Table 3-539. Function gptimer_trigger_adc_compare_shadow_enable.....	386
Table 3-540. Function gptimer_trigger_adc_compare_shadow_disable .....	387
Table 3-541. Function gptimer_trigger_adc_adsm_enable.....	388
Table 3-542. Function gptimer_trigger_adc_adsm_disable.....	388
Table 3-543. Function gptimer_trigger_adc_adsm_select.....	389
Table 3-544. Function gptimer_trigger_adc_skipping_config.....	390
Table 3-545. Function gptimer_trigger_adc_skipping_time_select .....	391
Table 3-546. Function gptimer_trigger_adc_skipping_counter_read .....	392
Table 3-547. Function gptimer_additional_interrupt_skipping_config.....	392
Table 3-548. Function gptimer_additional_interrupt_skipping_time_select .....	393
Table 3-549. Function gptimer_additional_interrupt_skipping_counter_read .....	395
Table 3-550. Function gptimer_flow_interrupt_skipping_source_select .....	396
Table 3-551. Function gptimer_flow_interrupt_skipping_link_enable.....	396
Table 3-552. Function gptimer_flow_interrupt_skipping_link_disable.....	397
Table 3-553. Function gptimer_flow_interrupt_skipping_num_config.....	398
Table 3-554. Function gptimer_flow_interrupt_skipping_counter_read .....	399
Table 3-555. Function gptimer_write_chxval_register_config .....	399
Table 3-556. Function gptimer_flag_get.....	400
Table 3-557. Function gptimer_flag_clear.....	401
Table 3-558. Function gptimer_interrupt_enable.....	402
Table 3-559. Function gptimer_interrupt_disable.....	403
Table 3-560. Function gptimer_interrupt_flag_get .....	404
Table 3-561. Function gptimer_interrupt_flag_clear .....	405
Table 3-562. GTOC registers.....	406
Table 3-563. GTOC firmware function.....	406
Table 3-564. Function gtoc_deinit .....	407
Table 3-565. Function gtoc_output_closing_request_enable .....	407
Table 3-566. Function gtoc_gptimer_trigger_status_get .....	408
Table 3-567. Function gtoc_software_request_generate .....	409



Table 3-568. Function gtoc_software_request_stop.....	409
Table 3-569. Function gtoc_input_detection_mode_select .....	410
Table 3-570. Function gtoc_input_polarity_config.....	410
Table 3-571. Function gtoc_input_polarity_config.....	411
Table 3-572. Function gtoc_digital_filter_disable.....	412
Table 3-573. Function gtoc_digital_filter_config .....	412
Table 3-574. Function gtoc_output_closing_request_mask.....	413
Table 3-575. Function gtoc_register_write_enable .....	414
Table 3-576. Function gtoc_register_write_disable .....	415
Table 3-577. Function gtoc_extended_closing_request_enable .....	415
Table 3-578. Function gtoc_extended_closing_request_disable .....	416
Table 3-579. Function gtoc_extended_closing_request_config .....	416
Table 3-580. Function gtoc_extended_closing_request_mask.....	417
Table 3-581. Function gtoc_flag_get.....	418
Table 3-582. Function gtoc_flag_clear.....	419
Table 3-583. Function gtoc_interrupt_flag_get .....	419
Table 3-584. Function gtoc_interrupt_flag_clear .....	420
Table 3-585. I2C Registers .....	421
Table 3-586. I2C firmware function.....	421
Table 3-587. Enum i2c_interrupt_flag_enum .....	423
Table 3-588. Function i2c_deinit .....	423
Table 3-589. Function i2c_timing_config .....	424
Table 3-590. Function i2c_digital_noise_filter_config .....	424
Table 3-591. Function i2c_master_clock_config .....	425
Table 3-592. Function i2c_master_addressing .....	426
Table 3-593. Function i2c_address10_header_enable.....	426
Table 3-594. Function i2c_address10_header_disable.....	427
Table 3-595. Function i2c_address10_enable .....	427
Table 3-596. Function i2c_address10_disable .....	428
Table 3-597. Function i2c_automatic_end_enable .....	428
Table 3-598. Function i2c_automatic_end_disable .....	429
Table 3-599. Function i2c_slave_response_to_gcall_enable .....	429
Table 3-600. Function i2c_slave_response_to_gcall_disable .....	430
Table 3-601. Function i2c_stretch_scl_low_enable .....	430
Table 3-602. Function i2c_stretch_scl_low_disable .....	431
Table 3-603. Function i2c_address_config .....	431
Table 3-604. Function i2c_address_bit_compare_config .....	432
Table 3-605. Function i2c_address_disable .....	433
Table 3-606. Function i2c_second_address_config.....	433
Table 3-607. Function i2c_second_address_disable .....	434
Table 3-608. Function i2c_receved_address_get .....	435
Table 3-609. Function i2c_slave_byte_control_enable.....	435
Table 3-610. Function i2c_slave_byte_control_disable.....	436
Table 3-611. Function i2c_nack_enable .....	436

Table 3-612. Function i2c_wakeup_from_deepsleep_enable .....	437
Table 3-613. Function i2c_wakeup_from_deepsleep_disable .....	437
Table 3-614. Function i2c_enable .....	437
Table 3-615. Function i2c_disable .....	438
Table 3-616. Function i2c_start_on_bus .....	438
Table 3-617. Function i2c_stop_on_bus .....	439
Table 3-618. Function i2c_data_transmit .....	439
Table 3-619. Function i2c_data_receive .....	440
Table 3-620. Function i2c_reload_enable.....	440
Table 3-621. Function i2c_reload_disable .....	441
Table 3-622. Function i2c_transfer_byte_number_config.....	441
Table 3-623. Function i2c_dma_enable .....	442
Table 3-624. Function i2c_dma_disable .....	442
Table 3-625. Function i2c_pec_transfer .....	443
Table 3-626. Function i2c_pec_enable .....	443
Table 3-627. Function i2c_pec_disable .....	444
Table 3-628. Function i2c_pec_value_get.....	444
Table 3-629. Function i2c_smbus_mode_enable .....	445
Table 3-630. Function i2c_smbus_mode_disable .....	445
Table 3-631. Function i2c_smbus_alert_enable.....	446
Table 3-632. Function i2c_smbus_alert_disable.....	446
Table 3-633. Function i2c_smbus_default_addr_enable .....	447
Table 3-634. Function i2c_smbus_default_addr_disable.....	447
Table 3-635. Function i2c_smbus_host_addr_enable .....	448
Table 3-636. Function i2c_smbus_host_addr_disable .....	448
Table 3-637. Function i2c_extented_clock_timeout_enable.....	449
Table 3-638. Function i2c_extented_clock_timeout_disable.....	449
Table 3-639. Function i2c_clock_timeout_enable .....	449
Table 3-640. Function i2c_clock_timeout_disable .....	450
Table 3-641. Function i2c_bus_timeout_b_config.....	450
Table 3-642. Function i2c_bus_timeout_a_config .....	451
Table 3-643. Function i2c_idle_clock_timeout_config .....	451
Table 3-644. Function i2c_flag_get .....	452
Table 3-645. Function i2c_flag_clear .....	453
Table 3-646. Function i2c_interrupt_enable .....	454
Table 3-647. Function i2c_interrupt_disable .....	454
Table 3-648. Function i2c_interrupt_flag_get.....	455
Table 3-649. Function i2c_interrupt_flag_clear.....	456
Table 3-650. NVIC Registers .....	457
Table 3-651. SysTick Registers .....	457
Table 3-652. MISC firmware function .....	457
Table 3-653. IRQn_Type.....	458
Table 3-654. Function nvic_priority_group_set .....	460
Table 3-655. Function nvic_irq_enable.....	460



Table 3-656. Function nvic_irq_disable.....	461
Table 3-657. Function nvic_system_reset .....	461
Table 3-658. Function nvic_vector_table_set.....	462
Table 3-659. Function system_lowpower_set .....	462
Table 3-660. Function system_lowpower_reset.....	463
Table 3-661. Function systick_clksource_set .....	464
Table 3-662. PMU Registers.....	464
Table 3-663. PMU firmware function .....	465
Table 3-664. Function pmu_deinit .....	465
Table 3-665. Function pmu_to_sleepmode .....	466
Table 3-666. Function pmu_to_deepsleepmode .....	466
Table 3-667. Function pmu_to_standbymode .....	467
Table 3-668. Function pmu_deepsleepmode_vcore_output.....	467
Table 3-669. Function pmu_wakeup_pin_enable .....	468
Table 3-670. Function pmu_wakeup_pin_disable .....	468
Table 3-671. Function pmu_lvd_enable.....	469
Table 3-672. Function pmu_lvd_disable.....	469
Table 3-673. Function pmu_lvd_interrupt_select .....	470
Table 3-674. Function pmu_lvd_interrupt_type .....	471
Table 3-675. Function pmu_lvd_interrupt_reset_enable .....	471
Table 3-676. Function pmu_lvd_interrupt_reset_disable .....	472
Table 3-677. Function pmu_lvd_output_enable .....	472
Table 3-678. Function pmu_lvd_output_disable .....	473
Table 3-679. Function pmu_lvd_mode_select.....	473
Table 3-680. Function pmu_lvd_reset_select.....	474
Table 3-681. Function pmu_lvd_level_select .....	475
Table 3-682. Function pmu_lvd_digital_filter_enable .....	475
Table 3-683. Function pmu_lvd_digital_filter_disable .....	476
Table 3-684. Function pmu_lvd_sample_clock_select .....	476
Table 3-685. Function pmu_flag_get.....	477
Table 3-686. Function pmu_flag_clear.....	478
Table 3-687. POC registers .....	478
Table 3-688. POC firmware function .....	479
Table 3-689. Structure poc_request_struct.....	480
Table 3-690. Structure poc_complementary_detection_struct .....	480
Table 3-691. Enum poc_pin_enum .....	480
Table 3-692. Enum cmp_output_enum .....	481
Table 3-693. Enum target_timer_enum .....	481
Table 3-694. Enum poc_interrupt_enum .....	481
Table 3-695. Function poc_deinit .....	481
Table 3-696. Function poc_input_detection_config .....	482
Table 3-697. Function poc_input_dreq_status_config .....	484
Table 3-698. Function poc_input_polarity_config .....	484
Table 3-699. Function poc_sys_fault_dreq_status_config .....	485

Table 3-700. Function poc_software_request_generate .....	486
Table 3-701. Function poc_software_request_stop .....	486
Table 3-702. Function poc_complementary_detection_struct_para_init .....	487
Table 3-703. Function poc_timer0_complementary_detection_config .....	487
Table 3-704. Function poc_timer7_complementary_detection_config .....	488
Table 3-705. Function poc_timer0_output_disable_mode_select .....	488
Table 3-706. Function poc_timer7_output_disable_mode_select .....	490
Table 3-707. Function poc_timer1_output_disable_mode_select .....	491
Table 3-708. Function poc_timer2_output_disable_mode_select .....	492
Table 3-709. Function poc_gptimer0_output_disable_mode_select .....	493
Table 3-710. Function poc_gptimer1_output_disable_mode_select .....	494
Table 3-711. Function poc_request_struct_para_init .....	495
Table 3-712. Function poc_request_select .....	495
Table 3-713. Function poc_cmp_dreq_status_config .....	496
Table 3-714. Function poc_cmp_dreq_status_extended_config .....	497
Table 3-715. Function poc_input_detection_mask .....	498
Table 3-716. Function poc_cmp_output_detection_mask .....	499
Table 3-717. Function poc_flag_get .....	501
Table 3-718. Function poc_flag_clear .....	502
Table 3-719. Function poc_interrupt_enable .....	503
Table 3-720. Function poc_interrupt_disable .....	504
Table 3-721. Function poc_interrupt_flag_get .....	504
Table 3-722. Function poc_interrupt_flag_clear .....	505
Table 3-723. RCU Registers .....	506
Table 3-724. RCU firmware function .....	506
Table 3-725. Enum rcu_periph_enum .....	507
Table 3-726. Enum rcu_periph_reset_enum .....	508
Table 3-727. Enum rcu_periph_sleep_enum .....	509
Table 3-728. Enum rcu_flag_enum .....	510
Table 3-729. Enum rcu_int_flag_enum .....	510
Table 3-730. Enum rcu_int_flag_clear_enum .....	511
Table 3-731. Enum rcu_int_enum .....	511
Table 3-732. Enum rcu_osci_type_enum .....	511
Table 3-733. Enum rcu_clock_freq_enum .....	511
Table 3-734. Function rcu_deinit .....	512
Table 3-735. Function rcu_periph_clock_enable .....	512
Table 3-736. Function rcu_periph_clock_disable .....	513
Table 3-737. Function rcu_periph_clock_sleep_enable .....	514
Table 3-738. Function rcu_periph_clock_sleep_disable .....	515
Table 3-739. Function rcu_periph_reset_enable .....	515
Table 3-740. Function rcu_periph_reset_disable .....	517
Table 3-741. Function rcu_system_clock_source_config .....	518
Table 3-742. Function rcu_system_clock_source_get .....	518
Table 3-743. Function rcu_ahb_clock_config .....	519

Table 3-744. Function rcu_apb1_clock_config .....	519
Table 3-745. Function rcu_apb2_clock_config .....	520
Table 3-746. Function rcu_ckout_config .....	521
Table 3-747. Function rcu_pll_config .....	522
Table 3-748. Function rcu_system_reset_enable .....	522
Table 3-749. Function rcu_system_reset_disable .....	523
Table 3-750. Function rcu_adc_clock_config.....	524
Table 3-751. Function rcu_i2c_clock_source_config .....	524
Table 3-752. Function rcu_osc_stab_wait .....	525
Table 3-753. Function rcu_osc_on .....	526
Table 3-754. Function rcu_osc_off.....	526
Table 3-755. Function rcu_osc_bypass_mode_enable .....	527
Table 3-756. Function rcu_osc_bypass_mode_disable .....	527
Table 3-757. Function rcu_hxtal_frequency_scale_select .....	528
Table 3-758. Function rcu_irc32m_adjust_value_set.....	528
Table 3-759. Function rcu_hxtal_clock_monitor_enable .....	529
Table 3-760. Function rcu_hxtal_clock_monitor_disable .....	529
Table 3-761. Function rcu_pll_clock_monitor_enable .....	530
Table 3-762. Function rcu_pll_clock_monitor_disable.....	530
Table 3-763. Function rcu_clock_freq_get.....	530
Table 3-764. Function rcu_flag_get.....	531
Table 3-765. Function rcu_all_reset_flag_clear .....	532
Table 3-766. Function rcu_interrupt_flag_get .....	533
Table 3-767. Function rcu_interrupt_flag_clear .....	533
Table 3-768. Function rcu_interrupt_enable.....	534
Table 3-769. Function rcu_interrupt_disable.....	535
Table 3-770. SPI Registers .....	536
Table 3-771. SPI firmware function .....	536
Table 3-772. spi_parameter_struct.....	537
Table 3-773. Function spi_deinit.....	537
Table 3-774. Function spi_struct_para_init .....	538
Table 3-775. Function spi_init .....	538
Table 3-776. Function spi_enable.....	539
Table 3-777. Function spi_disable .....	540
Table 3-778. Function spi_nss_output_enable .....	540
Table 3-779. Function spi_nss_output_disable .....	540
Table 3-780. Function spi_nss_internal_high .....	541
Table 3-781. Function spi_nss_internal_low .....	541
Table 3-782. Function spi_dma_enable .....	542
Table 3-783. Function spi_dma_disable.....	542
Table 3-784. Function spi_data_frame_format_config .....	543
Table 3-785. Function spi_data_transmit.....	544
Table 3-786. Function spi_data_receive.....	544
Table 3-787. Function spi_bidirectional_transfer_config.....	544

Table 3-788. Function spi_format_error_clear .....	545
Table 3-789. Function spi_crc_polynomial_set .....	546
Table 3-790. Function spi_crc_polynomial_get .....	546
Table 3-791. Function spi_crc_on .....	547
Table 3-792. Function spi_crc_off .....	547
Table 3-793. Function spi_crc_next .....	547
Table 3-794. Function spi_crc_get .....	548
Table 3-795. Function spi_crc_error_clear .....	549
Table 3-796. Function spi_ti_mode_enable .....	549
Table 3-797. Function spi_ti_mode_disable .....	549
Table 3-798. Function spi_nssp_mode_enable.....	550
Table 3-799. Function spi_nssp_mode_disable.....	550
Table 3-800. Function spi_quad_enable.....	551
Table 3-801. Function spi_quad_disable.....	551
Table 3-802. Function spi_quad_write_enable.....	552
Table 3-803. Function spi_quad_read_enable.....	552
Table 3-804. Function spi_flag_get .....	553
Table 3-805. Function spi_interrupt_enable .....	553
Table 3-806. Function spi_interrupt_disable .....	554
Table 3-807. Function spi_interrupt_flag_get.....	555
Table 3-808. SVPWM Registers .....	556
Table 3-809. SVPWM firmware function .....	556
Table 3-810. Structure svpwm_parameter_struct.....	556
Table 3-811. Function svpwm_deinit.....	556
Table 3-812. Function svpwm_init.....	557
Table 3-813. Function svpwm_enable .....	557
Table 3-814. Function svpwm_flag_get.....	558
Table 3-815. Function svpwm_alpha_beta_write.....	559
Table 3-816. Function svpwm_ta_tb_tc_read.....	559
Table 3-817. Function svpwm_sector_read .....	560
Table 3-818. SYSCFG Registers.....	560
Table 3-819. SYSCFG firmware function .....	561
Table 3-820. Enum exti_gpio_enum .....	562
Table 3-821. Enum syscfg_interrupt_enum .....	563
Table 3-822. Enum syscfg_adcs_m_enum.....	564
Table 3-823. Enum timer_channel_input_enum.....	565
Table 3-824. Enum ck_cmp_sel_enum .....	566
Table 3-825. Function syscfg_deinit .....	566
Table 3-826. Function syscfg_i2c_fast_mode_plus_enable .....	566
Table 3-827. Function syscfg_i2c_fast_mode_plus_disable .....	567
Table 3-828. Function syscfg_exti_line_config.....	567
Table 3-829. Function syscfg_timer_input_source_select .....	568
Table 3-830. Function syscfg_lockup_enable .....	568
Table 3-831. syscfg_sram_remap_set .....	569

Table 3-832. Function syscfg_sram_remap_size_get.....	570
Table 3-833. syscfg_ck_cmp_sel_set .....	570
Table 3-834. Function syscfg_ck_cmp_sel_get .....	571
Table 3-835. syscfg_register_write_enable .....	571
Table 3-836. syscfg_register_write_disable .....	572
Table 3-837. syscfg_adc_signal_monitor_select.....	572
Table 3-838. syscfg_adc_signal_monitor_output_enable.....	573
Table 3-839. syscfg_adc_signal_monitor_output_disable.....	574
Table 3-840. syscfg_boot_mode_get .....	574
Table 3-841. Function syscfg_sram_ecc_error_address_get .....	575
Table 3-842. Function syscfg_sram_ecc_error_bit_get .....	575
Table 3-843. Function syscfg_interrupt_enable.....	576
Table 3-844. Function syscfg_interrupt_disable.....	576
Table 3-845. Function syscfg_interrupt_flag_get .....	577
Table 3-846. Function syscfg_interrupt_flag_clear .....	578
Table 3-847. TIMERx Registers .....	579
Table 3-848. TIMERx firmware function.....	580
Table 3-849. Structure timer_parameter_struct .....	584
Table 3-850. Structure timer_break_parameter_struct.....	584
Table 3-851. Structure timer_oc_parameter_struct.....	585
Table 3-852. Structure timer_omc_parameter_struct .....	585
Table 3-853. Structure timer_ic_parameter_struct.....	585
Table 3-854. Function timer_deinit.....	586
Table 3-855. Function timer_struct_para_init.....	586
Table 3-856. Function timer_init .....	587
Table 3-857. Function timer_enable .....	588
Table 3-858. Function timer_disable .....	588
Table 3-859. Function timer_auto_reload_shadow_enable .....	589
Table 3-860. Function timer_auto_reload_shadow_disable .....	589
Table 3-861. Function timer_update_event_enable .....	590
Table 3-862. Function timer_update_event_disable .....	590
Table 3-863. Function timer_counter_alignment .....	591
Table 3-864. Function timer_counter_up_direction .....	591
Table 3-865. Function timer_counter_down_direction .....	592
Table 3-866. Function timer_upif_backup_enable.....	592
Table 3-867. Function timer_upif_backup_disable.....	593
Table 3-868. Function timer_flow_interrupt_source_select .....	593
Table 3-869. Function timer_update_source_select .....	594
Table 3-870. Function timer_external_input_source_select.....	595
Table 3-871. Function timer_prescaler_config.....	595
Table 3-872. Function timer_update_repetition_value_config .....	596
Table 3-873. Function timer_flow_interrupt_repetition_value_config .....	597
Table 3-874. Function timer_flow_interrupt_repetition_value_reload .....	597
Table 3-875. Function timer_autoreload_value_config .....	598

Table 3-876. Function timer_autoreload_value_read .....	598
Table 3-877. Function timer_counter_value_config .....	599
Table 3-878. Function timer_counter_read .....	599
Table 3-879. Function timer_prescaler_read .....	600
Table 3-880. Function timer_single_pulse_mode_config .....	600
Table 3-881. Function timer_update_source_config .....	601
Table 3-882. Function timer_channel_control_shadow_config .....	602
Table 3-883. Function timer_ocpre_clr_source_select .....	602
Table 3-884. Function timer_ocpre_clr_int_source_select .....	603
Table 3-885. Function timer_channel_composite_asymmetric_pwm_level_config .....	604
Table 3-886. Function timer_dma_enable .....	605
Table 3-887. Function timer_dma_disable .....	605
Table 3-888. Function timer_channel_dma_request_source_select .....	606
Table 3-889. Function timer_dma_transfer_config .....	607
Table 3-890. Function timer_event_software_generate .....	609
Table 3-891. Function timer_break_struct_para_init .....	611
Table 3-892. Function timer_break_config .....	611
Table 3-893. Function timer_break_enable .....	612
Table 3-894. Function timer_break_disable .....	613
Table 3-895. Function timer_automatic_output_enable .....	613
Table 3-896. Function timer_automatic_output_disable .....	614
Table 3-897. Function timer_primary_output_config .....	614
Table 3-898. Function timer_channel_output_struct_para_init .....	615
Table 3-899. Function timer_channel_output_config .....	615
Table 3-900. Function timer_channel_output_mode_config .....	616
Table 3-901. Function timer_channel_output_pulse_value_config .....	618
Table 3-902. Function timer_channel_output_shadow_config .....	619
Table 3-903. Function timer_channel_output_compare_fast_config .....	620
Table 3-904. Function timer_channel_output_clear_config .....	621
Table 3-905. Function timer_channel_output_polarity_config .....	622
Table 3-906. Function timer_channel_complementary_output_polarity_config .....	623
Table 3-907. Function timer_channel_output_state_config .....	623
Table 3-908. Function timer_channel_complementary_output_state_config .....	624
Table 3-909. Function timer_channel_input_struct_para_init .....	625
Table 3-910. Function timer_input_capture_config .....	626
Table 3-911. Function timer_channel_input_capture_prescaler_config .....	627
Table 3-912. Function timer_channel_capture_value_register_read .....	628
Table 3-913. Function timer_input_pwm_capture_config .....	628
Table 3-914. Function timer_hall_mode_config .....	629
Table 3-915. Function timer_multi_mode_channel_output_parameter_struct_init .....	630
Table 3-916. Function timer_multi_mode_channel_output_config .....	630
Table 3-917. Function timer_multi_mode_channel_mode_config .....	631
Table 3-918. Function timer_input_trigger_source_select .....	632
Table 3-919. Function timer_master_output_trigger_source_select .....	633



Table 3-920. Function timer_slave_mode_select .....	634
Table 3-921. Function timer_master_slave_mode_config .....	636
Table 3-922. Function timer_external_trigger_config .....	636
Table 3-923. Function timer_quadrature_decoder_mode_config .....	637
Table 3-924. Function timer_decoder_mode_config .....	638
Table 3-925. Function timer_internal_clock_config .....	639
Table 3-926. Function timer_internal_trigger_as_external_clock_config .....	640
Table 3-927. Function timer_external_trigger_as_external_clock_config .....	641
Table 3-928. Function timer_slave_mode_input_config .....	642
Table 3-929. Function timer_external_clock_mode0_config .....	643
Table 3-930. Function timer_external_clock_mode1_config .....	644
Table 3-931. Function timer_external_clock_mode1_disable .....	645
Table 3-932. Function timer_write_chxval_register_config .....	645
Table 3-933. Function timer_output_value_selection_config .....	646
Table 3-934. Function timer_output_match_pulse_select .....	646
Table 3-935. Function timer_channel_composite_pwm_pulse_config .....	647
Table 3-936. Function timer_channel_asymmetric_pwm_pulse_config .....	648
Table 3-937. Function timer_channel_additional_compare_value_config .....	649
Table 3-938. Function timer_channel_additional_compare_value_read .....	650
Table 3-939. Function timer_channel_additional_output_shadow_config .....	650
Table 3-940. Function timer_break_external_input_enable .....	651
Table 3-941. Function timer_break_cmp_enable .....	652
Table 3-942. Function timer_break_external_input_disable .....	652
Table 3-943. Function timer_break_cmp_disable .....	653
Table 3-944. Function timer_break_external_input_polarity_config .....	653
Table 3-945. Function timer_break_cmp_polarity_config .....	654
Table 3-946. Function timer_break_auto_recover_event_select .....	655
Table 3-947. Function timer_trigger_adc_compare_enable .....	655
Table 3-948. Function timer_trigger_adc_compare_disable .....	656
Table 3-949. Function timer_trigger_adc_flow_enable .....	657
Table 3-950. Function timer_trigger_adc_flow_disable .....	658
Table 3-951. Function timer_trigger_adc_compare_value_config .....	658
Table 3-952. Function timer_trigger_adc_repetition_value_config .....	659
Table 3-953. Function timer_trigger_adc_repetition_decrement_select .....	660
Table 3-954. timer_trigger_adc_repetition_value_reload .....	660
Table 3-955. Function timer_trigger_adc_compare_value_shadow_enable .....	661
Table 3-956. Function timer_trigger_adc_compare_value_shadow_disable .....	662
Table 3-957. Function timer_trigger_adc_monitor_config .....	662
Table 3-958. Function timer_register_update_event_select .....	664
Table 3-959. Function timer_flag_get .....	666
Table 3-960. Function timer_flag_clear .....	667
Table 3-961. Function timer_interrupt_enable .....	668
Table 3-962. Function timer_interrupt_disable .....	669
Table 3-963. Function timer_interrupt_flag_get .....	670

Table 3-964. Function timer_interrupt_flag_clear.....	672
Table 3-965. TMU Registers .....	673
Table 3-966. TMU firmware function .....	673
Table 3-967. Structure tmu_parameter_struct.....	674
Table 3-968. Function tmu_deinit .....	674
Table 3-969. Function tmu_struct_para_init.....	675
Table 3-970. Function tmu_init.....	675
Table 3-971. Function tmu_dma_read_enable .....	676
Table 3-972. Function tmu_dma_read_disable .....	677
Table 3-973. Function tmu_dma_write_enable .....	677
Table 3-974. Function tmu_dma_write_disable .....	678
Table 3-975. Function tmu_one_q31_write .....	678
Table 3-976. Function tmu_two_q31_write .....	679
Table 3-977. Function tmu_two_q15_write .....	679
Table 3-978. Function tmu_one_f32_write .....	680
Table 3-979. Function tmu_two_f32_write .....	680
Table 3-980. Function tmu_one_q31_read .....	681
Table 3-981. Function tmu_two_q31_read .....	682
Table 3-982. Function tmu_two_q15_read .....	682
Table 3-983. Function tmu_one_f32_read .....	683
Table 3-984. Function tmu_two_f32_read .....	683
Table 3-985. Function tmu_flag_get.....	684
Table 3-986. Function tmu_flag_clear.....	684
Table 3-987. Function tmu_interrupt_enable.....	685
Table 3-988. Function tmu_interrupt_disable.....	685
Table 3-989. Function tmu_interrupt_flag_get .....	686
Table 3-990. Function tmu_interrupt_flag_clear .....	686
Table 3-991. UART Registers.....	687
Table 3-992. UART firmware function .....	688
Table 3-993. Enum uart_flag_enum.....	689
Table 3-994. Enum uart_interrupt_flag_enum .....	689
Table 3-995. Enum uart_interrupt_enum.....	689
Table 3-996. Enum uart_invert_enum .....	690
Table 3-997. Function uart_deinit .....	690
Table 3-998. Function uart_baudrate_set.....	690
Table 3-999. Function uart_parity_config .....	691
Table 3-1000. Function uart_word_length_set.....	692
Table 3-1001. Function uart_stop_bit_set.....	692
Table 3-1002. Function uart_enable .....	693
Table 3-1003. Function uart_disable .....	693
Table 3-1004. Function uart_transmit_config.....	694
Table 3-1005. Function uart_receive_config .....	694
Table 3-1006. Function uart_data_first_config.....	695
Table 3-1007. Function uart_invert_config .....	696



Table 3-1008. Function <code>uart_oversample_config</code> .....	696
Table 3-1009. Function <code>uart_sample_bit_config</code> .....	697
Table 3-1010. Function <code>uart_data_transmit</code> .....	697
Table 3-1011. Function <code>uart_data_receive</code> .....	698
Table 3-1012. Function <code>uart_address_config</code> .....	698
Table 3-1013. Function <code>uart_mute_mode_enable</code> .....	699
Table 3-1014. Function <code>uart_mute_mode_disable</code> .....	699
Table 3-1015. Function <code>uart_mute_mode_wakeup_config</code> .....	700
Table 3-1016. Function <code>uart_lin_mode_enable</code> .....	701
Table 3-1017. Function <code>uart_lin_mode_disable</code> .....	701
Table 3-1018. Function <code>uart_lin_break_dection_length_config</code> .....	702
Table 3-1019. Function <code>uart_halfduplex_enable</code> .....	702
Table 3-1020. Function <code>uart_halfduplex_disable</code> .....	703
Table 3-1021. Function <code>uart_halfduplex_disable</code> .....	703
Table 3-1022. Function <code>uart_irda_mode_enable</code> .....	704
Table 3-1023. Function <code>uart_irda_mode_disable</code> .....	704
Table 3-1024. Function <code>uart_prescaler_config</code> .....	705
Table 3-1025. Function <code>uart_irda_lowpower_config</code> .....	705
Table 3-1026. Function <code>uart_parity_check_coherence_config</code> .....	706
Table 3-1027. Function <code>uart_dma_receive_config</code> .....	706
Table 3-1028. Function <code>uart_dma_transmit_config</code> .....	707
Table 3-1029. Function <code>uart_flag_get</code> .....	708
Table 3-1030. Function <code>uart_flag_clear</code> .....	708
Table 3-1031. Function <code>uart_interrupt_enable</code> .....	709
Table 3-1032. Function <code>uart_interrupt_disable</code> .....	709
Table 3-1033. Function <code>uart_interrupt_flag_get</code> .....	710
Table 3-1034. Function <code>uart_interrupt_flag_clear</code> .....	711
Table 3-1035. WWDGT Registers .....	712
Table 3-1036. WWDGT firmware function .....	712
Table 3-1037. Structure <code>wwdgt_cfg_parameter_struct</code> .....	712
Table 3-1038. Function <code>wwdgt_deinit</code> .....	712
Table 3-1039. Function <code>wwdgt_counter_update</code> .....	713
Table 3-1040. Function <code>wwdgt_struct_para_init</code> .....	713
Table 3-1041. Function <code>wwdgt_cfg_init</code> .....	714
Table 3-1042. Function <code>wwdgt_interrupt_enable</code> .....	714
Table 3-1043. Function <code>wwdgt_flag_get</code> .....	715
Table 3-1044. Function <code>wwdgt_flag_clear</code> .....	716
Table 4-1. Revision history .....	717

## 1. Introduction

This manual introduces firmware library of GD32M53x 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 GD32M53x 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
CFMU	Clock frequency measurement unit
CMP	Comparator
CPTIMER	Compare timer

Peripherals	Descriptions
CPTIMERW	Compare timer W
CRC	Cyclic redundancy checks management unit
DAC	Digital-to-analog converter
DBG	Debug
DMA	Direct memory access controller
EVIC	Event interconnection unit
EXTI	Interrupt/event controller
FMC	Flash memory controller
FWDGT	Free watchdog timer
GPIO	General-purpose and alternate-function I/Os
GPTIMER	General purpose timer
GTOC	GPTIMER Output Controller
I2C	Inter-integrated circuit interface
MISC	Nested Vectored Interrupt Controller
PMU	Power management unit
POC	Port output controller
RCU	Reset and clock unit
SPI/I2S	Serial peripheral interface
SVPWM	Space vector pulse width modulation
SYSCFG	System configuration
TIMER	TIMER
TMU	Trigonometric Math Unit
UART	Universal asynchronous receiver / transmitter
WWDGT	Window watchdog timer

### 1.1.2. Naming rules

The firmware library naming rules are shown as below:

- The peripherals are shortened in XXX format, such as: ADC. More shorten information of peripherals refer to [Peripherals](#);
- The name of sourcefile and header file are started with “gd32m53x\_”, such as: gd32m53x\_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

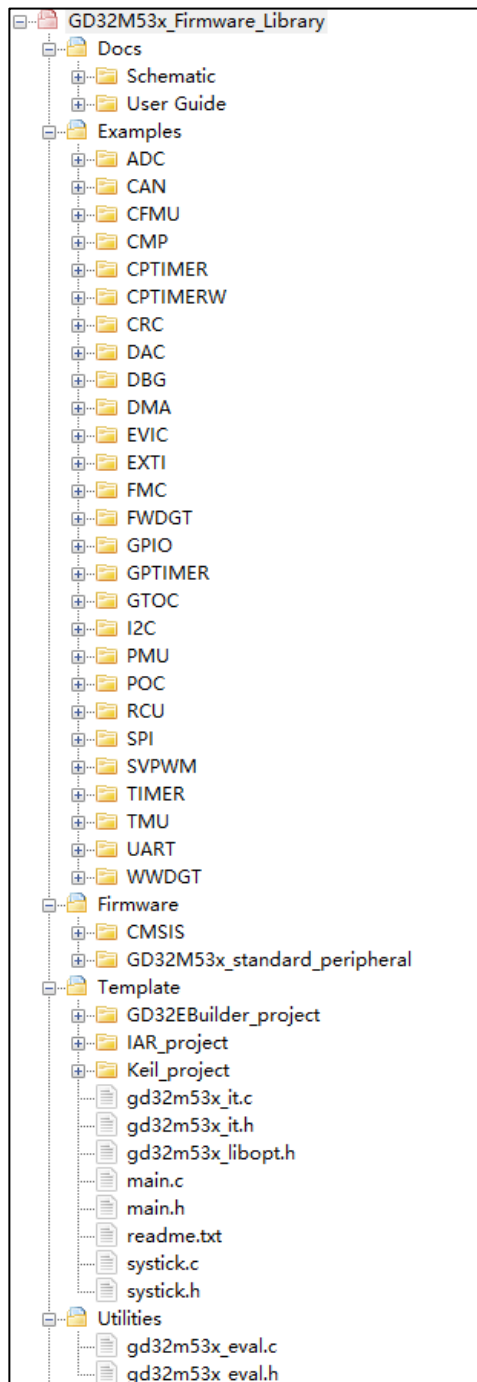
case.

## 2. Firmware Library Overview

### 2.1. File Structure of Firmware Library

GD32M53x\_Firmware\_Library, the file structure is shown as below:

**Figure 2-1. File structure of firmware library of GD32M53x**



### 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;
- `gd32m53x_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);
- `gd32m53x_it.c`: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- `GD32M53x_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 GD32M53x and system configuration file;
- GD32M53x\_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.

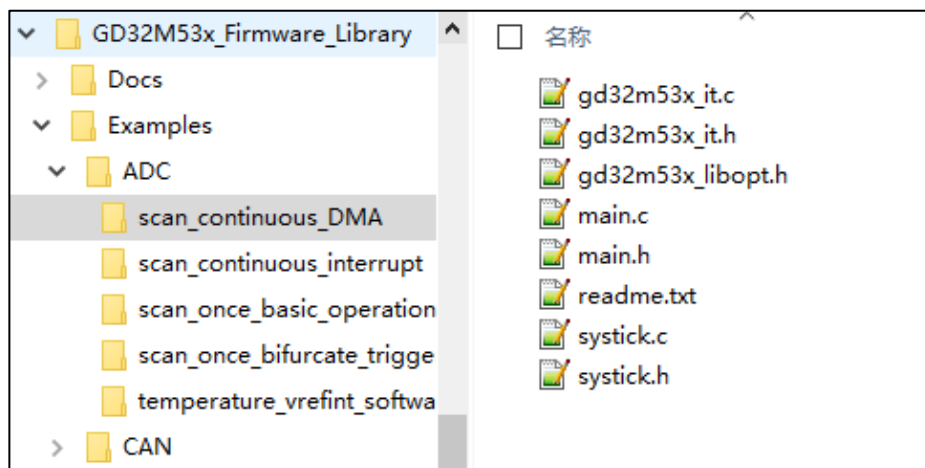
### 2.1.3. Template Folder

Template folder includes a simple demo of how to use LED, how to print by UART and use key to control, (IAR\_project is run in IAR, Keil\_project is run in Keil5, and GD32EBuilder\_project is run in GD32EmbeddedBuilder). User can use the project template to compile the formware examples, the steps are shown as below:

#### Select files

Open "Examples" folder, select the module to be tested, such as ADC, open "ADC" folder, select an example of ADC, such as "scan\_continuous\_DMA", shown as below:

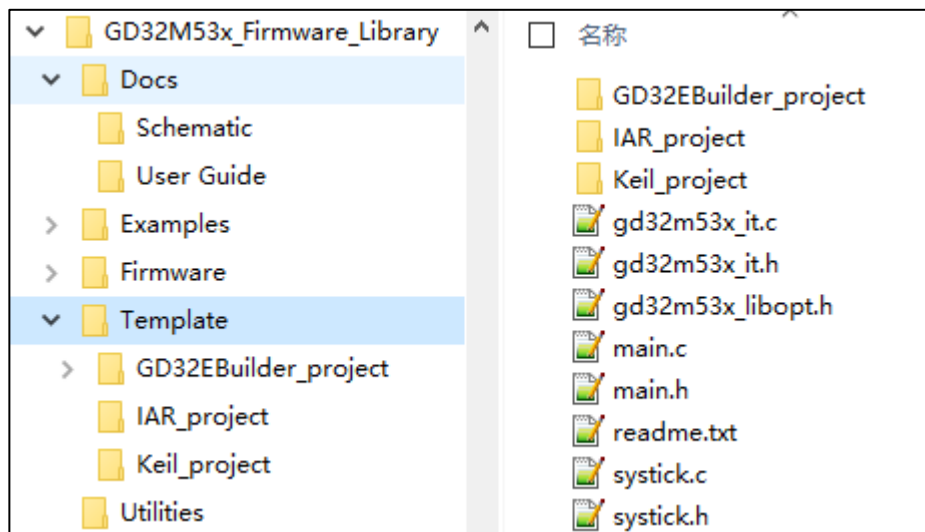
**Figure 2-2. Select peripheral example files**



## Copy files

Open "Template" folder, keep the folders of "IAR\_project", "Keil\_project", and "GD32EBuilder\_project", and delete the other files, then copy all the files in "scan\_continuous\_DMA" folder to the "Template" subfolder, shown as below:

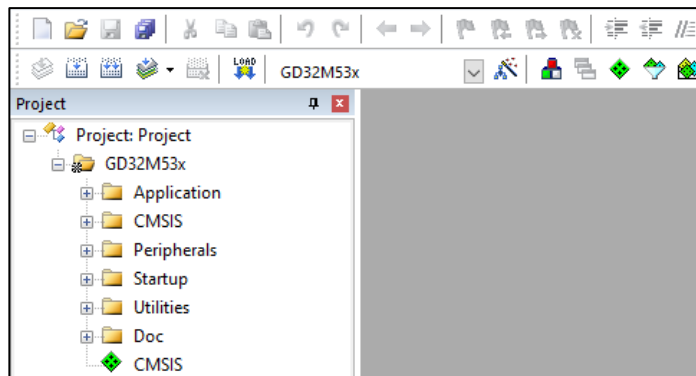
**Figure 2-3. Copy the peripheral example files**



## Open a project

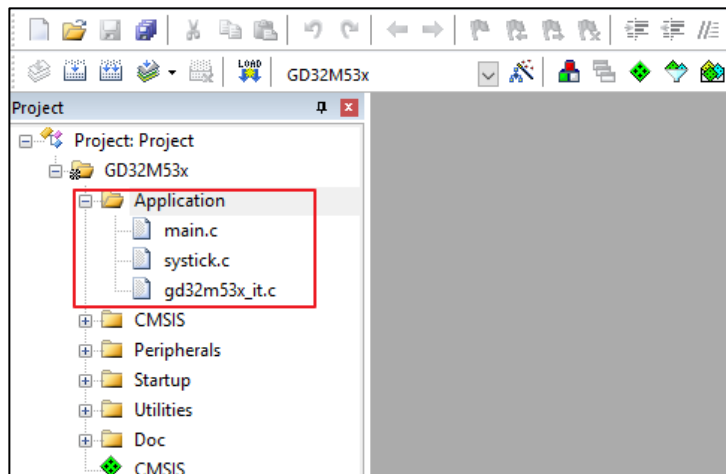
GD provides project in Keil, IAR and GD32EmbeddedBuilder, users can open project in different IDEs according to their need, such as "Keil\_project", open \Template\Keil\_project\Project.uvprojx, shown as below:

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:

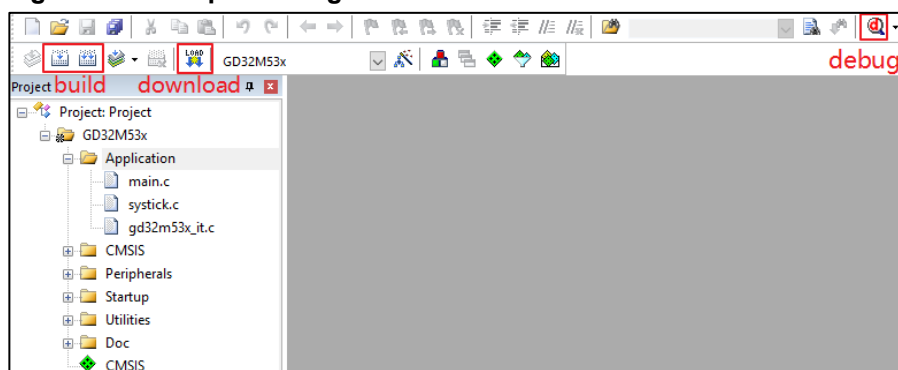
Figure 2-5. Configure project files



## 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:

- gd32m53x\_eval.h is related header file of the evaluation board about running the firmware examples;
- gd32m53x\_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
gd32m53x_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.
gd32m53x_it.h	Header file, including all the prototypes of interrupt service routines.
gd32m53x_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.
gd32m53x_xxx.h	The header file of peripheral xxx, including functions about peripheral xxx, and the variables used for functions.
gd32m53x_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

The 12-bit ADC is an analog-to-digital converter using the successive approximation method. The ADC registers are listed in chapter [3.2.1](#) the ADC firmware functions are introduced in chapter [3.2.2](#).

#### 3.2.1. Descriptions of Peripheral registers

ADC registers are listed in the table shown as below:

**Table 3-2. ADC Registers**

<b>Registers</b>	<b>Descriptions</b>
ADC_EOCCTL	software EOC control register
ADC_CTL0	control register 0
ADC_GPCGF0	group config register 0
ADC_GPCGF1	group config register 1
ADC_CHSEL0	channel selection register 0
ADC_CHSEL1	channel selection register 1
ADC_SAMPR0	sample time register 0
ADC_SAMPR1	sample time register 1

ADC_SAMPR2	sample time register 2, only for ADC2
ADC_CHPRIR0	channel priority register 0
ADC_CHPRIR1	channel priority register 1
ADC_ADDT0	channel addition value config register 0
ADC_ADDT1	channel addition value config register 1, only for ADC2
ADC_SDDATA	self-diagnosis data register
ADC_GP1BIDATA	Group_pri1 bifurcate data register
ADC_GP1BIDATA1	Group_pri1 bifurcate data register 1
ADC_GP1BIDATA2	Group_pri1 bifurcate data register 2
ADC_GP2BIDATA	Group_pri2 bifurcate data register
ADC_GP2BIDATA1	Group_pri2 bifurcate data register 1
ADC_GP2BIDATA2	Group_pri2 bifurcate data register 2
ADC_GP3BIDATA	Group_pri3 bifurcate data register
ADC_GP3BIDATA1	Group_pri3 bifurcate data register 1
ADC_GP3BIDATA2	Group_pri3 bifurcate data register 2
ADC_GP4BIDATA	Group_pri4 bifurcate data register
ADC_GP4BIDATA1	Group_pri4 bifurcate data register 1
ADC_GP4BIDATA2	Group_pri4 bifurcate data register 2
ADC_GP1DMAR	Group_pri1 data DMA register
ADC_GP2DMAR	Group_pri2 data DMA register
ADC_GP3DMAR	Group_pri3 data DMA register
ADC_GP4DMAR	Group_pri4 data DMA register
ADC_CH0DATA	channel 0 data register
ADC_CH1DATA	channel 1 data register
ADC_CH2DATA	channel 2 data register
ADC_CH3DATA	channel 3 data register
ADC_CH4DATA	channel 4 data register
ADC_CH5DATA	channel 5 data register
ADC_CH6DATA	channel 6 data register
ADC_CH7DATA	channel 7 data register, only for ADC2
ADC_CH8DATA	channel 8 data register, only for ADC2
ADC_CH9DATA	channel 9 data register, only for ADC2
ADC_TEMPDATA	temperature sensor channel data register, only for ADC2
ADC_VINTDATA	internal reference voltage channel data register, only for ADC2
ADC_WDCTL	watchdog control register
ADC_WDATHOLD	watchdog A threshold register
ADC_WDBTHOLD	watchdog B threshold register
ADC_WDACHCFG	watchdog A channel config register
ADC_WDACHSTAT	watchdog A channel status register
ADC_CTL1	control register 1
ADC_DMACTL	DMA control register
ADC_BITRGCTL	bifurcate trigger control register

### 3.2.2. Descriptions of Peripheral functions

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

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

Function name	Function description
adc_deinit	reset ADC
adc_enable	enable ADC interface
adc_disable	disable ADC interface
adc_data_alignment_config	configure ADC data alignment
adc_resolution_config	configure ADC resolution
adc_self_diagnosis_enable	enable self-diagnosis
adc_self_diagnosis_disable	disable self-diagnosis
adc_self_diagnosis_mode_config	config self-diagnosis
adc_disconnect_detect_mode_config	configure disconnection detection assist mode
adc_disconnect_detect_period_config	configure disconnect detect period
adc_data_auto_clear_enable	enable automatic clearing data registers
adc_data_auto_clear_disable	disable automatic clearing data registers
adc_data_auto_set_enable	enable automatic setting data registers
adc_data_auto_set_disable	disable automatic setting data registers
adc_sample_hold_channel_config	configure ADC sample-and-hold channel
adc_sh_sample_time_config	configure ADC sample time for sample-and-hold circuits
adc_sh_hold_time_config	configure ADC hold time for sample-and-hold circuits
adc_sh_constant_sampling_mode_enable	enable ADC constant sampling mode for sample-and-hold circuits
adc_sh_constant_sampling_mode_disable	disable ADC constant sampling mode for sample-and-hold circuits
adc_sh_constant_sampling_start	software start sample-and-hold circuit in constant sampling mode
adc_sh_constant_sampling_stop	software end sample-and-hold circuit in constant sampling mode
adc_watchdog_enable	enable ADC analog watchdog
adc_watchdog_disable	disable ADC analog watchdog
adc_watchdog_a_channel_config	configure ADC analog watchdog A channel
adc_watchdog_a_channel_deselect	deselect ADC analog watchdog A channel
adc_watchdog_b_channel_config	configure ADC analog watchdog B channel
adc_watchdog_a_threshold_config	configure ADC analog watchdog A threshold
adc_watchdog_b_threshold_config	configure ADC analog watchdog B threshold
adc_watchdog_window_mode_enable	enable ADC analog watchdog window function
adc_watchdog_window_mode_disable	disable ADC analog watchdog window function
adc_watchdog_complex_condition_config	configure ADC analog watchdog complex

	conditions
adc_oversample_channel_config	configure ADC oversample channel
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_group_scan_mode_config	configure the ADC scan mode
adc_group_priority_control_enable	enable group priority control
adc_group_priority_control_disable	disable group priority control
adc_group_pri3_enable	enable A/D conversion operation for Group_pri3
adc_group_pri3_disable	disable A/D conversion operation for Group_pri3
adc_group_pri4_enable	enable A/D conversion operation for Group_pri4
adc_group_pri4_disable	disable A/D conversion operation for Group_pri4
adc_group_lowest_priority_continuous_enable	enable lowest-priority group scan continuous
adc_group_lowest_priority_continuous_disable	disable lowest-priority group scan continuous
adc_group_restart_enable	enable low-priority group restart
adc_group_restart_disable	disable low-priority group restart
adc_group_channel_config	select ADC channel for Group_pri x
adc_group_channel_deselect	deselect ADC group channel
adc_group_end_conversion_round_config	config end of Group_prix conversion round counts
adc_group_external_trigger_enable	enable ADC external trigger
adc_group_extern_trigger_edge_config	config ADC external trigger edge
adc_group_external_trigger_disable	disable ADC external trigger
adc_group_synchronous_trigger_enable	enable ADC synchronous trigger
adc_group_synchronous_trigger_source_config	configure ADC synchronous trigger source
adc_group_asynchronous_trigger_enable	enable ADC asynchronous trigger
adc_group_software_trigger_enable	enable ADC software trigger
adc_group_software_end_conversion	software end conversion of all groups
adc_group_bifurcate_mode_enable	enable ADC bifurcate trigger mode for Group_pri x
adc_group_bifurcate_mode_disable	disable ADC bifurcate trigger mode for Group_pri x
adc_group_bifurcate_channel_select	ADC bifurcate trigger channel select
adc_group_bifurcate_extend_trigger_select	extend bifurcate trigger source select
adc_group_bifurcate_trigger_restart_enable	enable restore of the next trigger during the current A/D conversion round
adc_group_bifurcate_trigger_restart_disable	disable restore of the next trigger during the current A/D conversion round
adc_group_dma_mode_enable	enable DMA request
adc_group_dma_mode_disable	disable DMA request
adc_group_dma_request_after_last_enable	when DMAENx=1, the DMA engine issues request at end of conversion of Group_prix
adc_group_dma_request_after_last_disable	the DMA engine is disabled after the end of transfer signal from DMA controller is detected
adc_group_dma_overrun_detect_enable	enable dma overrun detect

adc_group_dma_overrun_detect_disable	disable dma overrun detect
adc_channel_priority_config	channel priority config
adc_channel_sample_time_config	configure ADC channel sample time
adc_evic_link_signal_event_config	select which event as the EVIC link signal
adc_group_evic_link_signal_source	select EVIC link signal source for Group_pri x
adc_channel_data_read	read ADC channel data register
adc_self_diagnosis_data_read	read ADC self-diagnosis data register
adc_self_diagnosis_status_read	read ADC self-diagnosis data register
adc_bifurcate_data_read	read ADC bifurcate data register
adc_flag_get	get the ADC flag
adc_flag_clear	clear the ADC flag
adc_interrupt_enable	enable ADC interrupt
adc_interrupt_disable	disable ADC interrupt
adc_interrupt_flag_get	get ADC interrupt flag
adc_interrupt_flag_clear	clear ADC interrupt flag

### Enum adc\_group\_select\_enum

Table 3-4. Enum adc\_group\_select\_enum

Member name	Function description
ADC_GROUP_PRI4	select Group_pri4
ADC_GROUP_PRI3	select Group_pri3
ADC_GROUP_PRI2	select Group_pri2
ADC_GROUP_PRI1	select Group_pri1

### Enum adc\_channel\_select\_enum

Table 3-5. Enum adc\_channel\_select\_enum

Member name	Function description
ADC_CHANNEL_IN00	select channel 0 (ADCx_IN00)
ADC_CHANNEL_IN01	select channel 1 (ADCx_IN01)
ADC_CHANNEL_IN02	select channel 2 (ADCx_IN02)
ADC_CHANNEL_IN03	select channel 3 (ADCx_IN03)
ADC_CHANNEL_IN04	select channel 4 (ADCx_IN04)
ADC_CHANNEL_IN05	select channel 5 (ADCx_IN05)
ADC_CHANNEL_IN06	select channel 6 (ADCx_IN06)
ADC_CHANNEL_IN07	select channel 7 (ADCx_IN07) , only for ADC2
ADC_CHANNEL_IN08	select channel 8 (ADCx_IN08) , only for ADC2
ADC_CHANNEL_IN09	select channel 9 (ADCx_IN09) , only for ADC2
ADC_CHANNEL_TEMPERATURE	select temperature sensor channel, only for ADC2
ADC_CHANNEL_VINT	select internal reference voltage, only for ADC2
ADC_CHANNEL_ALL	select all channel

## Enum adc\_bifurcate\_data\_enum

Table 3-6. Enum adc\_bifurcate\_data\_enum

Member name	Function description
ADC_BIFURCATE_DATA	select Group_pri x bifurcate data register (ADC_GPxBIDATA)
ADC_EXTENDED_BIFURCATE_DATA_1	select Group_pri x bifurcate data register 1 (ADC_GPxBIDATA1)
ADC_EXTENDED_BIFURCATE_DATA_2	select Group_pri x bifurcate data register 2 (ADC_GPxBIDATA2)

## Enum adc\_disc\_detect\_mode\_enum

Table 3-7. Enum adc\_disc\_detect\_mode\_enum

Member name	Function description
ADC_DISCHARGE_MODE	select discharge mode
ADC_PRECHARGE_MODE	select precharge mode
ADC_PRECHARGE_DISCHARGE_MODE_DISABLE	disable precharge and discharge mode

## Enum adc\_self\_diag\_mode\_enum

Table 3-8. Enum adc\_self\_diag\_mode\_enum

Member name	Function description
ADC_SELF_DIAGNOSIS_MODE_ROTATION	select rotation mode for self-diagnosis voltage
ADC_SELF_DIAGNOSIS_MODE_FIXED	select fixed mode for self-diagnosis voltage

## Enum adc\_self\_diag\_fixed\_voltage\_enum

Table 3-9. Enum adc\_self\_diag\_fixed\_voltage\_enum

Member name	Function description
ADC_SELF_DIAG_0_V	uses the voltage of 0 V for self-diagnosis
ADC_SELF_DIAG_1_2_VREF	uses the voltage of reference power supply $V_{REFP}/2$ for self-diagnosis
ADC_SELF_DIAG_VREF	uses the voltage of reference power supply $V_{REFP}$ for self-diagnosis

## Enum adc\_restart\_channel\_enum

Table 3-10. Enum adc\_restart\_channel\_enum

Member name	Function description
ADC_RESTART_ON_BEGINNING	rescan the lower priority group scanning from initial channel
ADC_RESTART_ON_BREAKING	rescan the lower priority group scanning from the last aborted channel

## Enum adc\_watchdog\_select\_enum

Table 3-11. Enum adc\_watchdog\_select\_enum

Member name	Function description
ADC_WATCHDOG_A	select analog watchdog A
ADC_WATCHDOG_B	select analog watchdog B
ADC_WATCHDOG_A_B	select analog watchdog A and watchdog B

## Enum adc\_watchdog\_compare\_condition\_enum

Table 3-12. Enum adc\_watchdog\_compare\_condition\_enum

Member name	Function description
ADC_OUT_WINDOW	<p>When the WINEN is 0, watchdog compare conditions: WDLT[15:0] value &gt; A/D-converted value.</p> <p>When the WINEN is 1, watchdog compare conditions: A/D-converted value &lt; WDLT[15:0] value or WDHT[15:0] &lt; A/D-converted value</p>
ADC_IN_WINDOW	<p>When the WINEN is 0, watchdog compare conditions: WDLT[15:0] value &lt; A/D-converted value.</p> <p>When the WINEN is 1, watchdog compare conditions: WDLT[15:0] value &lt; A/D-converted value &lt; WDHT[15:0] value</p>

## Enum adc\_oversample\_mode\_enum

Table 3-13. Enum adc\_oversample\_mode\_enum

Member name	Function description
ADC_ACCUMULATION_MODE	accumulation data mode
ADC_AVERAGE_MODE	average data mode

## Enum adc\_interrupt\_enum

Table 3-14. Enum adc\_interrupt\_enum

Member name	Function description
ADC_INT_EOC1RF	interrupt for EOC1RF
ADC_INT_EOC2RF	interrupt for EOC2RF
ADC_INT_EOC3RF	interrupt for EOC3RF
ADC_INT_EOC4RF	interrupt for EOC4RF
ADC_INT_WDA_CHSTAT	interrupt for watchdog A
ADC_INT_WDBEF	interrupt for watchdog B
ADC_INT_GP1OVRF	DMA overflow detect interrupt of Group_pri1
ADC_INT_GP2OVRF	DMA overflow detect interrupt of Group_pri2
ADC_INT_GP3OVRF	DMA overflow detect interrupt of Group_pri3



ADC_INT_GP4OVRF	DMA overflow detect interrupt of Group_pri4
-----------------	---

### Enum adc\_interrupt\_flag\_enum

**Table 3-15. Enum adc\_interrupt\_flag\_enum**

Member name	Function description
ADC_INT_FLAG_EOC1RF	interrupt for EOC1RF
ADC_INT_FLAG_EOC2RF	interrupt for EOC2RF
ADC_INT_FLAG_EOC3RF	interrupt for EOC3RF
ADC_INT_FLAG_EOC4RF	interrupt for EOC4RF
ADC_INT_FLAG_WDA_CHSTAT	interrupt for watchdog A
ADC_INT_FLAG_WDBEF	interrupt for watchdog B
ADC_INT_FLAG_GP1OVRF	DMA overflow detect interrupt of Group_pri1
ADC_INT_FLAG_GP2OVRF	DMA overflow detect interrupt of Group_pri2
ADC_INT_FLAG_GP3OVRF	DMA overflow detect interrupt of Group_pri3
ADC_INT_FLAG_GP4OVRF	DMA overflow detect interrupt of Group_pri4

### Enum adc\_event\_flag\_enum

**Table 3-16. Enum adc\_event\_flag\_enum**

Member name	Function description
ADC_FLAG_EOC1F	end of Group_pri1 conversion flag
ADC_FLAG_EOC1RF	end of Group_pri1 conversion round flag
ADC_FLAG_EOC2F	end of Group_pri2 conversion flag
ADC_FLAG_EOC2RF	end of Group_pri2 conversion round flag
ADC_FLAG_EOC3F	end of Group_pri3 conversion flag
ADC_FLAG_EOC3RF	end of Group_pri3 conversion round flag
ADC_FLAG_EOC4F	end of Group_pri4 conversion flag
ADC_FLAG_EOC4RF	end of Group_pri4 conversion round flag
ADC_FLAG_PROC	A/D conversion process flag
ADC_FLAG_WDAMF	analog watchdog A compare monitor flag
ADC_FLAG_WDBMF	analog watchdog B compare monitor flag
ADC_FLAG_WDABMF	analog watchdog A/B complex compare monitor flag
ADC_FLAG_GP1OVRF	DMA overflow flag of Group_pri1
ADC_FLAG_GP2OVRF	DMA overflow flag of Group_pri2
ADC_FLAG_GP3OVRF	DMA overflow flag of Group_pri3
ADC_FLAG_GP4OVRF	DMA overflow flag of Group_pri4
ADC_FLAG_WDBEF	analog watchdog B event flag
ADC_FLAG_WDA_CH0CMPF	channel ADCx_IN00 compare status in watchdog A
ADC_FLAG_WDA_CH1CMPF	channel ADCx_IN01 compare status in watchdog A
ADC_FLAG_WDA_CH2CMPF	channel ADCx_IN02 compare status in watchdog A

	A
ADC_FLAG_WDA_CH3CMPF	channel ADCx_IN03 compare status in watchdog A
ADC_FLAG_WDA_CH4CMPF	channel ADCx_IN04 compare status in watchdog A
ADC_FLAG_WDA_CH5CMPF	channel ADCx_IN05 compare status in watchdog A
ADC_FLAG_WDA_CH6CMPF	channel ADCx_IN06 compare status in watchdog A
ADC_FLAG_WDA_CH7CMPF	channel ADCx_IN07 compare status in watchdog A
ADC_FLAG_WDA_CH8CMPF	channel ADCx_IN08 compare status in watchdog A
ADC_FLAG_WDA_CH9CMPF	channel ADCx_IN09 compare status in watchdog A
ADC_FLAG_WDA_TEMPCMPF	temperature sensor channel compare status in watchdog A
ADC_FLAG_WDA_VINTCMPF	internal reference voltage channel compare status in watchdog A
ADC_FLAG_WDA_ALL_CHCMPF	all channel compare status in watchdog A

### Enum adc\_evic\_link\_source\_enum

Table 3-17. Enum adc\_evic\_link\_source\_enum

Member name	Function description
ADC_EVIC_LINK_SOURCE_EOCRF_FLAG	EVIC trigger signal selects EOCxRF (x=1,2,3,4)
ADC_EVIC_LINK_SOURCE_EOCF_FLAG	EVIC trigger signal selects EOCxF (x=1,2,3,4)

### adc\_deinit

The description of adc\_deinit is shown as below:

Table 3-18. 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=0,2)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### adc\_enable

The description of adc\_enable is shown as below:

**Table 3-19. 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>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### adc\_disable

The description of adc\_disable is shown as below:

**Table 3-20. 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>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### adc\_data\_alignment\_config

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

**Table 3-21. 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 ADC data alignment
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>data_alignment</b>	data alignment select
ADC_DATAALIGN_RIGHT	right alignment (LSB)
ADC_DATAALIGN_LEFT	left alignment (MSB)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### adc\_resolution\_config

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

**Table 3-22. Function adc\_resolution\_config**

<b>Function name</b>	adc_resolution_config
<b>Function prototype</b>	void adc_resolution_config(uint32_t adc_periph, uint32_t resolution)
<b>Function descriptions</b>	configure ADC resolution
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC resolution */
adc_resolution_config(ADC0, ADC_RESOLUTION_10B);
```

### adc\_self\_diagnosis\_enable

The description of adc\_self\_diagnosis\_enable is shown as below:

**Table 3-23. Function adc\_self\_diagnosis\_enable**

<b>Function name</b>	adc_self_diagnosis_enable
<b>Function prototype</b>	void adc_self_diagnosis_enable(uint32_t adc_periph)
<b>Function descriptions</b>	enable self-diagnosis
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable self-diagnosis */
adc_self_diagnosis_enable(ADC0);
```

## adc\_self\_diagnosis\_disable

The description of adc\_self\_diagnosis\_disable is shown as below:

**Table 3-24. Function adc\_self\_diagnosis\_disable**

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

Example:

```
/* enable self-diagnosis */
```

```
adc_self_diagnosis_disable(ADC0);
```

## adc\_self\_diagnosis\_mode\_config

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

**Table 3-25. Function adc\_self\_diagnosis\_mode\_config**

<b>Function name</b>	adc_self_diagnosis_mode_config
<b>Function prototype</b>	void adc_self_diagnosis_mode_config(uint32_t adc_periph, adc_self_diag_mode_enum mode, adc_self_diag_fixed_voltage_enum voltage_select)
<b>Function descriptions</b>	config self-diagnosis
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>mode</b>	self-diagnosis mode select. The members can refer to <a href="#"><u>Enum adc_self_diag_mode_enum</u></a> .
<b>Input parameter{in}</b>	
<b>voltage_select</b>	configure fix voltage for self-diagnosis conversion. The members can refer to <a href="#"><u>Enum adc_self_diag_fixed_voltage_enum</u></a> .
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* config self-diagnosis */
```

```
adc_self_diagnosis_mode_config (ADC0, ADC_SELF_DIAGNOSIS_MODE_FIXED,
ADC_SELF_DIAG_1_2_VREF);
```

### adc\_disconnect\_detect\_mode\_config

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

**Table 3-26. Function adc\_disconnect\_detect\_mode\_config**

<b>Function name</b>	adc_disconnect_detect_mode_config
<b>Function prototype</b>	void adc_disconnect_detect_mode_config(uint32_t adc_periph, adc_disc_detect_mode_enum mode)
<b>Function descriptions</b>	configure disconnection detection assist mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>mode</b>	disconnection detection assist mode select. The members can refer to <a href="#">Enum adc_disc_detect_mode_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure disconnection detection assist mode */
```

```
adc_disconnect_detect_mode_config (ADC0, ADC_DISCHARGE_MODE);
```

### adc\_disconnect\_detect\_period\_config

The description of adc\_disconnect\_detect\_period\_config is shown as below:

**Table 3-27. Function adc\_disconnect\_detect\_period\_config**

<b>Function name</b>	adc_disconnect_detect_period_config
<b>Function prototype</b>	void adc_disconnect_detect_period_config(uint32_t adc_periph, uint32_t period)
<b>Function descriptions</b>	configure disconnect detect period

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>period</b>	disconnect detect period(0x0~0xF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure disconnect detect period */
adc_disconnect_detect_period_config (ADC0, 0x3);
```

### adc\_data\_auto\_clear\_enable

The description of adc\_data\_auto\_clear\_enable is shown as below:

**Table 3-28. Function adc\_data\_auto\_clear\_enable**

<b>Function name</b>	adc_data_auto_clear_enable
<b>Function prototype</b>	void adc_data_auto_clear_enable(uint32_t adc_periph)
<b>Function descriptions</b>	enable automatic clearing data registers
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable automatic clearing data registers */
adc_data_auto_clear_enable (ADC0);
```

### adc\_data\_auto\_clear\_disable

The description of adc\_data\_auto\_clear\_disable is shown as below:

**Table 3-29. Function adc\_data\_auto\_clear\_disable**

<b>Function name</b>	adc_data_auto_clear_disable
----------------------	-----------------------------



<b>Function prototype</b>	void adc_data_auto_clear_disable(uint32_t adc_periph)
<b>Function descriptions</b>	disable automatic clearing data registers
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable automatic clearing data registers */
```

```
adc_data_auto_clear_disable (ADC0);
```

### adc\_data\_auto\_set\_enable

The description of adc\_data\_auto\_set\_enable is shown as below:

**Table 3-30. Function adc\_data\_auto\_set\_enable**

<b>Function name</b>	adc_data_auto_set_enable
<b>Function prototype</b>	void adc_data_auto_set_enable(uint32_t adc_periph)
<b>Function descriptions</b>	enable automatic setting data registers
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable automatic setting data registers */
```

```
adc_data_auto_set_enable (ADC0);
```

### adc\_data\_auto\_set\_disable

The description of adc\_data\_auto\_set\_disable is shown as below:

**Table 3-31. Function adc\_data\_auto\_set\_disable**

<b>Function name</b>	adc_data_auto_set_disable
----------------------	---------------------------

<b>Function prototype</b>	void adc_data_auto_set_disable(uint32_t adc_periph)
<b>Function descriptions</b>	disable automatic setting data registers
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable automatic setting data registers */
```

```
adc_data_auto_set_disable (ADC0);
```

### adc\_sample\_hold\_channel\_config

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

**Table 3-32. Function adc\_sample\_hold\_channel\_config**

<b>Function name</b>	adc_sample_hold_channel_config
<b>Function prototype</b>	void adc_sample_hold_channel_config(uint32_t channel)
<b>Function descriptions</b>	configure ADC sample-and-hold channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channel</b>	sample-and-hold channel select
<i>ADC_SH_CHANNEL_I N00</i>	use ADC0_IN00 channel-dedicated sample-and-hold circuits
<i>ADC_SH_CHANNEL_I N01</i>	use ADC0_IN01 channel-dedicated sample-and-hold circuits
<i>ADC_SH_CHANNEL_I N02</i>	use ADC0_IN02 channel-dedicated sample-and-hold circuits
<b>Input parameter{in}</b>	
<b>ctl</b>	control status
<i>ENABLE</i>	enable channel-dedicated sample-and-hold circuits
<i>DISABLE</i>	disable channel-dedicated sample-and-hold circuits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC sample-and-hold channel */
```

```
adc_sample_hold_channel_config (ADC_SH_CHANNEL_IN00);
```

### adc\_sh\_sample\_time\_config

The description of adc\_sh\_sample\_time\_config is shown as below:

**Table 3-33. Function adc\_sh\_sample\_time\_config**

<b>Function name</b>	adc_sh_sample_time_config
<b>Function prototype</b>	void adc_sh_sample_time_config(uint8_t sample_time)
<b>Function descriptions</b>	configure ADC sample time for sample-and-hold circuits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sample_time</b>	0x08~0xFF. The recommended config value should not be less than 8.
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC sample time for sample-and-hold circuits */
```

```
adc_sh_sample_time_config (12);
```

### adc\_sh\_hold\_time\_config

The description of adc\_sh\_hold\_time\_config is shown as below:

**Table 3-34. Function adc\_sh\_hold\_time\_config**

<b>Function name</b>	adc_sh_hold_time_config
<b>Function prototype</b>	void adc_sh_hold_time_config(uint8_t hold_time)
<b>Function descriptions</b>	configure ADC hold time for sample-and-hold circuits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sample_time</b>	0x8~0xF. The recommended config value should not be less than 8.
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC hold time for sample-and-hold circuits */
```

```
adc_sh_hold_time_config (12);
```

## adc\_sh\_constant\_sampling\_mode\_enable

The description of adc\_sh\_constant\_sampling\_mode\_enable is shown as below:

**Table 3-35. Function adc\_sh\_constant\_sampling\_mode\_enable**

<b>Function name</b>	adc_sh_constant_sampling_mode_enable
<b>Function prototype</b>	void adc_sh_constant_sampling_mode_enable(void)
<b>Function descriptions</b>	enable ADC constant sampling mode for sample-and-hold circuits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ADC constant sampling mode for sample-and-hold circuits */
adc_sh_constant_sampling_mode_enable ();
```

## adc\_sh\_constant\_sampling\_mode\_disable

The description of adc\_sh\_constant\_sampling\_mode\_disable is shown as below:

**Table 3-36. Function adc\_sh\_constant\_sampling\_mode\_disable**

<b>Function name</b>	adc_sh_constant_sampling_mode_disable
<b>Function prototype</b>	void adc_sh_constant_sampling_mode_disable(void)
<b>Function descriptions</b>	disable ADC constant sampling mode for sample-and-hold circuits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC constant sampling mode for sample-and-hold circuits */
adc_sh_constant_sampling_mode_disable ();
```

## adc\_sh\_constant\_sampling\_start

The description of adc\_sh\_constant\_sampling\_start is shown as below:

Table 3-37. Function `adc_sh_constant_sampling_start`

Function name	<code>adc_sh_constant_sampling_start</code>
Function prototype	<code>void adc_sh_constant_sampling_start(void)</code>
Function descriptions	software start sample-and-hold circuit in constant sampling mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* software start sample-and-hold circuit in constant sampling mode */
adc_sh_constant_sampling_start ();
```

### `adc_sh_constant_sampling_stop`

The description of `adc_sh_constant_sampling_stop` is shown as below:

Table 3-38. Function `adc_sh_constant_sampling_stop`

Function name	<code>adc_sh_constant_sampling_stop</code>
Function prototype	<code>void adc_sh_constant_sampling_stop(void)</code>
Function descriptions	software end sample-and-hold circuit in constant sampling mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* software start sample-and-hold circuit in constant sampling mode */
adc_sh_constant_sampling_stop ();
```

### `adc_watchdog_enable`

The description of `adc_watchdog_enable` is shown as below:

Table 3-39. Function `adc_watchdog_enable`

Function name	<code>adc_watchdog_enable</code>
---------------	----------------------------------

<b>Function prototype</b>	void adc_watchdog_enable(uint32_t adc_periph, adc_watchdog_select_enum window)
<b>Function descriptions</b>	enable ADC analog watchdog
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>window</b>	the selected analog watchdog. The members can refer to <a href="#">Enum adc_watchdog_select_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_watchdog_enable (ADC0, ADC_WATCHDOG_A);
```

### adc\_watchdog\_disable

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

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

<b>Function name</b>	adc_watchdog_disable
<b>Function prototype</b>	void adc_watchdog_disable(uint32_t adc_periph, adc_watchdog_select_enum window)
<b>Function descriptions</b>	disable ADC analog watchdog
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>window</b>	the selected analog watchdog. The members can refer to <a href="#">Enum adc_watchdog_select_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disableADC analog watchdog */
```

adc\_watchdog\_disable (ADC0, ADC\_WATCHDOG\_A);

### adc\_watchdog\_a\_channel\_config

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

**Table 3-41. Function adc\_watchdog\_a\_channel\_config**

<b>Function name</b>	adc_watchdog_a_channel_config
<b>Function prototype</b>	void adc_watchdog_a_channel_config(uint32_t adc_periph, adc_channel_select_enum channel, adc_watchdog_compare_condition_enum compare_mode)
<b>Function descriptions</b>	configure ADC analog watchdog A channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	the selected ADC channel. The members can refer to <a href="#">Enum adc_channel_select_enum</a> .
<b>Input parameter{in}</b>	
<b>compare_mode</b>	watchdog compare conditions mode. The members can refer to <a href="#">Enum adc_watchdog_compare_condition_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_watchdog_a_channel_config (ADC0, ADC_CHANNEL_IN00, ADC_IN_WINDOW);
```

### adc\_watchdog\_a\_channel\_deselect

The description of adc\_watchdog\_a\_channel\_deselect is shown as below:

**Table 3-42. Function adc\_watchdog\_a\_channel\_deselect**

<b>Function name</b>	adc_watchdog_a_channel_deselect
<b>Function prototype</b>	void adc_watchdog_a_channel_deselect(uint32_t adc_periph, adc_channel_select_enum channel)
<b>Function descriptions</b>	deselect ADC analog watchdog A channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral

<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	the selected ADC channel. The members can refer to <a href="#">Enum <i>adc_watchdog_compare_condition_enum</i></a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deselect ADC analog watchdog A channel */
```

```
adc_watchdog_a_channel_deselect (ADC0, ADC_CHANNEL_IN00);
```

### **adc\_watchdog\_b\_channel\_config**

The description of `adc_watchdog_b_channel_config` is shown as below:

**Table 3-43. Function `adc_watchdog_b_channel_config`**

<b>Function name</b>	<code>adc_watchdog_b_channel_config</code>
<b>Function prototype</b>	<code>void adc_watchdog_b_channel_config(uint32_t adc_periph, adc_channel_select_enum channel, adc_watchdog_compare_condition_enum compare_mode)</code>
<b>Function descriptions</b>	configure ADC analog watchdog B channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	the selected ADC channel. The members can refer to <a href="#">Enum <i>adc_watchdog_compare_condition_enum</i></a> . <b>Note:</b> except for <code>ADC_CHANNEL_ALL</code> .
<b>Input parameter{in}</b>	
<b>compare_mode</b>	watchdog compare conditions mode. The members can refer to <a href="#">Enum <i>adc_watchdog_compare_condition_enum</i></a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_watchdog_b_channel_config (ADC0, ADC_CHANNEL_IN00, ADC_IN_WINDOW);
```



## adc\_watchdog\_a\_threshold\_config

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

**Table 3-44. Function adc\_watchdog\_a\_threshold\_config**

<b>Function name</b>	adc_watchdog_a_threshold_config
<b>Function prototype</b>	void adc_watchdog_a_threshold_config(uint32_t adc_periph, uint16_t low_threshold, uint16_t high_threshold)
<b>Function descriptions</b>	configure ADC analog watchdog A threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>low_threshold</b>	analog watchdog low threshold, 0x0000..0xFFFF
<b>Input parameter{in}</b>	
<b>high_threshold</b>	analog watchdog high threshold, 0x0000..0xFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC analog watchdog A threshold */
```

```
adc_watchdog_a_threshold_config (ADC0, 0x1F3F, 0x2FFF);
```

## adc\_watchdog\_b\_threshold\_config

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

**Table 3-45. Function adc\_watchdog\_b\_threshold\_config**

<b>Function name</b>	adc_watchdog_b_threshold_config
<b>Function prototype</b>	void adc_watchdog_b_threshold_config(uint32_t adc_periph, uint16_t low_threshold, uint16_t high_threshold)
<b>Function descriptions</b>	configure ADC analog watchdog B threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>low_threshold</b>	analog watchdog low threshold, 0x0000..0xFFFF
<b>Input parameter{in}</b>	

<b>high_threshold</b>	analog watchdog high threshold, 0x0000..0xFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC analog watchdog B threshold */
```

```
adc_watchdog_b_threshold_config (ADC0, 0x1F3F, 0x2FFF);
```

### adc\_watchdog\_window\_mode\_enable

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

**Table 3-46. Function adc\_watchdog\_window\_mode\_enable**

<b>Function name</b>	adc_watchdog_window_mode_enable
<b>Function prototype</b>	void adc_watchdog_window_mode_enable(uint32_t adc_periph)
<b>Function descriptions</b>	enable ADC analog watchdog window function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ADC analog watchdog window function */
```

```
adc_watchdog_window_mode_enable (ADC0);
```

### adc\_watchdog\_window\_mode\_disable

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

**Table 3-47. Function adc\_watchdog\_window\_mode\_disable**

<b>Function name</b>	adc_watchdog_window_mode_disable
<b>Function prototype</b>	void adc_watchdog_window_mode_disable(uint32_t adc_periph)
<b>Function descriptions</b>	disable ADC analog watchdog window function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral

<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC analog watchdog window function */
```

```
adc_watchdog_window_mode_disable (ADC0);
```

### adc\_watchdog\_complex\_condition\_config

The description of `adc_watchdog_complex_condition_config` is shown as below:

**Table 3-48. Function `adc_watchdog_complex_condition_config`**

<b>Function name</b>	<code>adc_watchdog_complex_condition_config</code>
<b>Function prototype</b>	<code>void adc_watchdog_complex_condition_config(uint32_t adc_periph, uint32_t mode)</code>
<b>Function descriptions</b>	configure ADC analog watchdog complex conditions
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>mode</b>	Watchdog A/B complex conditions configuration
<i>ADC_WATCHDOG_MATCH_A_OR_B</i>	(Watchdog A condition matched) OR (Watchdog B condition matched)
<i>ADC_WATCHDOG_MATCH_A_XOR_B</i>	(Watchdog A condition matched) XOR (Watchdog B condition matched)
<i>ADC_WATCHDOG_MATCH_A_AND_B</i>	(Watchdog A condition matched) AND (Watchdog B condition matched)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC analog watchdog complex conditions */
```

```
adc_watchdog_complex_condition_config (ADC0, ADC_WATCHDOG_MATCH_A_XOR_B);
```

### adc\_oversample\_channel\_config

The description of `adc_oversample_channel_config` is shown as below:

Table 3-49. Function `adc_oversample_channel_config`

Function name	<code>adc_oversample_channel_config</code>
Function prototype	<code>void adc_oversample_channel_config(uint32_t adc_periph, adc_channel_select_enum channel, uint32_t ratio)</code>
Function descriptions	configure ADC oversample channel
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_periph</code>	ADC peripheral
<code>ADCx(x=0,2)</code>	ADC peripheral selection
Input parameter{in}	
<code>channel</code>	the selected ADC channel. The members can refer to <a href="#">Enum <code>adc_watchdog_compare_condition_enum</code></a> . <b>Note:</b> except for <code>ADC_CHANNEL_ALL</code> .
Input parameter{in}	
<code>ratio</code>	ADC oversampling ratio
<code>ADC_OVERSAMPLING_RATIO_MUL_1</code>	oversampling ratio multiple 1
<code>ADC_OVERSAMPLING_RATIO_MUL_2</code>	oversampling ratio multiple 2
<code>ADC_OVERSAMPLING_RATIO_MUL_3</code>	oversampling ratio multiple 3
<code>ADC_OVERSAMPLING_RATIO_MUL_4</code>	oversampling ratio multiple 4
<code>ADC_OVERSAMPLING_RATIO_MUL_8</code>	oversampling ratio multiple 8
<code>ADC_OVERSAMPLING_RATIO_MUL_16</code>	oversampling ratio multiple 16
<code>ADC_OVERSAMPLING_RATIO_MUL_32</code>	oversampling ratio multiple 32
<code>ADC_OVERSAMPLING_RATIO_MUL_64</code>	oversampling ratio multiple 64
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ADC oversample channel */
```

```
adc_oversample_channel_config(ADC0, ADC_CHANNEL_IN00,
ADC_OVERSAMPLING_RATIO_MUL_2);
```

## adc\_oversample\_mode\_config

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

**Table 3-50. 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, adc_oversample_mode_enum mode)
<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=0,2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>mode</b>	A/D conversion data addition mode selection. The members can refer to <a href="#">Enum adc_oversample_mode_enum</a> .
ADC_ACCUMULATION_MODE	accumulation data mode
ADC_AVERAGE_MODE	average data mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC oversample mode */
adc_oversample_mode_config (ADC0, ADC_AVERAGE_MODE);
```

## adc\_oversample\_mode\_enable

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

**Table 3-51. Function adc\_oversample\_mode\_enable**

<b>Function name</b>	adc_oversample_mode_enable
<b>Function prototype</b>	void adc_oversample_mode_enable(uint32_t adc_periph)
<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
ADCx(x=0,2)	ADC peripheral selection
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

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

```
adc_oversample_mode_enable (ADC0);
```

### adc\_oversample\_mode\_disable

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

**Table 3-52. Function adc\_oversample\_mode\_disable**

<b>Function name</b>	adc_oversample_mode_disable
<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
<b>ADCx(x=0,2)</b>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC oversample mode */
```

```
adc_oversample_mode_disable (ADC0);
```

### adc\_group\_scan\_mode\_config

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

**Table 3-53. Function adc\_group\_scan\_mode\_config**

<b>Function name</b>	adc_group_scan_mode_config
<b>Function prototype</b>	void adc_group_scan_mode_config(uint32_t adc_periph, uint32_t scan_mode)
<b>Function descriptions</b>	configure the ADC scan mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<b>ADCx(x=0,2)</b>	ADC peripheral selection

Input parameter{in}	
<b>scan_mode</b>	ADC scan mode
<i>ADC_GROUP_PRI1_SCAN_ONCE</i>	Group_pri1 scan once mode
<i>ADC_GROUPS_SCAN</i>	Groups scan mode
<i>ADC_GROUP_PRI1_SCAN_CONTINUE</i>	Group_pri1 scan continuous mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the ADC scan mode */
```

```
adc_oversample_mode_disable (ADC0, ADC_GROUP_PRI1_SCAN_ONCE);
```

### adc\_group\_priority\_control\_enable

The description of adc\_group\_priority\_control\_enable is shown as below:

**Table 3-54. Function adc\_group\_priority\_control\_enable**

<b>Function name</b>	adc_group_priority_control_enable
<b>Function prototype</b>	void adc_group_priority_control_enable(uint32_t adc_periph)
<b>Function descriptions</b>	enable group priority control
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable group priority control */
```

```
adc_group_priority_control_enable (ADC0);
```

### adc\_group\_priority\_control\_disable

The description of adc\_group\_priority\_control\_disable is shown as below:

**Table 3-55. Function adc\_group\_priority\_control\_disable**

<b>Function name</b>	adc_group_priority_control_disable
----------------------	------------------------------------

<b>Function prototype</b>	void adc_group_priority_control_disable(uint32_t adc_periph)
<b>Function descriptions</b>	disable group priority control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<b>ADCx(x=0,2)</b>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable group priority control */
adc_group_priority_control_disable (ADC0);
```

### adc\_group\_pri3\_enable

The description of adc\_group\_pri3\_enable is shown as below:

**Table 3-56. Function adc\_group\_pri3\_enable**

<b>Function name</b>	adc_group_pri3_enable
<b>Function prototype</b>	void adc_group_pri3_enable(uint32_t adc_periph)
<b>Function descriptions</b>	enable A/D conversion operation for Group_pri3
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<b>ADCx(x=0,2)</b>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable A/D conversion operation for Group_pri3 */
adc_group_pri3_enable (ADC0);
```

### adc\_group\_pri3\_disable

The description of adc\_group\_pri3\_disable is shown as below:

**Table 3-57. Function adc\_group\_pri3\_disable**

<b>Function name</b>	adc_group_pri3_disable
----------------------	------------------------



<b>Function prototype</b>	void adc_group_pri3_disable(uint32_t adc_periph)
<b>Function descriptions</b>	disable A/D conversion operation for Group_pri3
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable A/D conversion operation for Group_pri3 */
```

```
adc_group_pri3_disable (ADC0);
```

### adc\_group\_pri4\_enable

The description of adc\_group\_pri4\_enable is shown as below:

**Table 3-58. Function adc\_group\_pri4\_enable**

<b>Function name</b>	adc_group_pri4_enable
<b>Function prototype</b>	void adc_group_pri4_enable(uint32_t adc_periph)
<b>Function descriptions</b>	enable A/D conversion operation for Group_pri4
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable A/D conversion operation for Group_pri4 */
```

```
adc_group_pri4_enable (ADC0);
```

### adc\_group\_pri4\_disable

The description of adc\_group\_pri4\_disable is shown as below:

**Table 3-59. Function adc\_group\_pri4\_disable**

<b>Function name</b>	adc_group_pri4_disable
----------------------	------------------------

<b>Function prototype</b>	void adc_group_pri4_disable(uint32_t adc_periph)
<b>Function descriptions</b>	disable A/D conversion operation for Group_pri4
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<b>ADCx(x=0,2)</b>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable A/D conversion operation for Group_pri4 */
```

```
adc_group_pri4_disable (ADC0);
```

### adc\_group\_lowest\_priority\_continuous\_enable

The description of adc\_group\_lowest\_priority\_continuous\_enable is shown as below:

**Table 3-60. Function adc\_group\_lowest\_priority\_continuous\_enable**

<b>Function name</b>	adc_group_lowest_priority_continuous_enable
<b>Function prototype</b>	void adc_group_lowest_priority_continuous_enable(uint32_t adc_periph)
<b>Function descriptions</b>	enable lowest-priority group scan continuous
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<b>ADCx(x=0,2)</b>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable lowest-priority group scan continuous */
```

```
adc_group_lowest_priority_continuous_enable (ADC0);
```

### adc\_group\_lowest\_priority\_continuous\_disable

The description of adc\_group\_lowest\_priority\_continuous\_disable is shown as below:

**Table 3-61. Function adc\_group\_lowest\_priority\_continuous\_disable**

<b>Function name</b>	adc_group_lowest_priority_continuous_disable
----------------------	--

<b>Function prototype</b>	void adc_group_lowest_priority_continuous_disable(uint32_t adc_periph)
<b>Function descriptions</b>	disable lowest-priority group scan continuous
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable lowest-priority group scan continuous */
```

```
adc_group_lowest_priority_continuous_disable (ADC0);
```

### adc\_group\_restart\_enable

The description of adc\_group\_restart\_enable is shown as below:

**Table 3-62. Function adc\_group\_restart\_enable**

<b>Function name</b>	adc_group_restart_enable
<b>Function prototype</b>	void adc_group_restart_enable(uint32_t adc_periph, adc_group_select_enum group, adc_restart_channel_enum restart_ch)
<b>Function descriptions</b>	enable low-priority group restart
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> . <b>Note:</b> except for <i>ADC_GROUP_PRI1</i> .
<b>Input parameter{in}</b>	
<b>restart_ch</b>	rescan channel select. The members can refer to <a href="#">Enum adc_restart_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable low-priority group restart */
```

adc\_group\_restart\_enable (ADC0, ADC\_GROUP\_PRI2, ADC\_RESTART\_ON\_BEGINNING);

### adc\_group\_restart\_disable

The description of adc\_group\_restart\_disable is shown as below:

**Table 3-63. Function adc\_group\_restart\_disable**

<b>Function name</b>	adc_group_restart_disable
<b>Function prototype</b>	void adc_group_restart_disable(uint32_t adc_periph, adc_group_select_enum group)
<b>Function descriptions</b>	disable low-priority group restart
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> . <b>Note:</b> except for <i>ADC_GROUP_PRI1</i> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable low-priority group restart */
```

```
adc_group_restart_disable (ADC0, ADC_GROUP_PRI2);
```

### adc\_group\_channel\_config

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

**Table 3-64. Function adc\_group\_channel\_config**

<b>Function name</b>	adc_group_channel_config
<b>Function prototype</b>	void adc_group_channel_config(uint32_t adc_periph, adc_group_select_enum group, adc_channel_select_enum channel, uint32_t sample_time)
<b>Function descriptions</b>	select ADC channel for Group_pri x
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	

group	select the group. The members can refer to <a href="#">Enum</a> <a href="#">adc_group_select_enum</a> .
Input parameter{in}	
channel	the selected ADC channel. The members can refer to <a href="#">Enum</a> <a href="#">adc_watchdog_compare_condition_enum</a> .
Input parameter{in}	
sample_time	0x02~0xFF. The sample time value 0x02 cycle
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select ADC channel for Group_pri x */
```

```
adc_group_channel_config (ADC0, ADC_GROUP_PRI2, ADC_CHANNEL_IN00, 0x04);
```

### adc\_group\_channel\_deselect

The description of adc\_group\_channel\_deselect is shown as below:

**Table 3-65. Function adc\_group\_channel\_deselect**

Function name	adc_group_channel_deselect
Function prototype	void adc_group_channel_deselect(uint32_t adc_periph, adc_group_select_enum group, adc_channel_select_enum channel)
Function descriptions	deselect ADC group channel
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
Input parameter{in}	
group	select the group. The members can refer to <a href="#">Enum</a> <a href="#">adc_group_select_enum</a> .
Input parameter{in}	
channel	the selected ADC channel. The members can refer to <a href="#">Enum</a> <a href="#">adc_watchdog_compare_condition_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deselect ADC group channel */
```

```
adc_group_channel_deselect (ADC0, ADC_GROUP_PRI2, ADC_CHANNEL_IN00);
```

### adc\_group\_end\_flag\_round\_config

The description of adc\_group\_end\_flag\_round\_config is shown as below:

**Table 3-66. Function adc\_group\_end\_flag\_round\_config**

<b>Function name</b>	adc_group_end_flag_round_config
<b>Function prototype</b>	void adc_group_end_flag_round_config(uint32_t adc_periph, adc_group_select_enum group, uint8_t num)
<b>Function descriptions</b>	config end of Group_prix conversion round counts
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> .
<b>Input parameter{in}</b>	
<b>num</b>	end of Group_prix conversion round counts, 0~7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config end of Group_prix conversion round counts */
```

```
adc_group_end_flag_round_config (ADC0, ADC_GROUP_PRI2, 3);
```

### adc\_group\_external\_trigger\_enable

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

**Table 3-67. Function adc\_group\_external\_trigger\_enable**

<b>Function name</b>	adc_group_external_trigger_enable
<b>Function prototype</b>	void adc_group_external_trigger_enable(uint32_t adc_periph, adc_group_select_enum group)
<b>Function descriptions</b>	enable ADC external trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	

<b>group</b>	select the group. The members can refer to <a href="#">Enum</a> <a href="#">adc_group_select_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ADC external trigger */
```

```
adc_group_external_trigger_enable (ADC0, ADC_GROUP_PRI2, 3);
```

### adc\_group\_extern\_trigger\_edge\_config

The description of adc\_group\_extern\_trigger\_edge\_config is shown as below:

**Table 3-68. Function adc\_group\_extern\_trigger\_edge\_config**

<b>Function name</b>	adc_group_extern_trigger_edge_config
<b>Function prototype</b>	void adc_group_extern_trigger_edge_config(uint32_t adc_periph, adc_group_select_enum group, uint32_t edge_sel)
<b>Function descriptions</b>	config ADC external trigger edge
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum</a> <a href="#">adc_group_select_enum</a> .
<b>Input parameter{in}</b>	
<b>edge_sel</b>	select trigger edge
<i>ADC_EXTERNAL_TRIGGER_RISING_EDGE</i>	Rising edge of external trigger enable
<i>ADC_EXTERNAL_TRIGGER_FALLING_EDGE</i>	Falling edge of external trigger enable
<i>ADC_EXTERNAL_TRIGGER_BOTH_EDGE</i>	Rising and falling edge of external trigger enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ADC external trigger */
```

```
adc_group_external_trigger_enable (ADC0, ADC_GROUP_PRI2,
ADC_EXTERNAL_TRIG_RISING_EDGE);
```

### adc\_group\_external\_trigger\_disable

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

**Table 3-69. Function adc\_group\_external\_trigger\_disable**

<b>Function name</b>	adc_group_external_trigger_disable
<b>Function prototype</b>	void adc_group_external_trigger_disable(uint32_t adc_periph, adc_group_select_enum group)
<b>Function descriptions</b>	disable ADC external trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC external trigger */
```

```
adc_group_external_trigger_disable (ADC0, ADC_GROUP_PRI2);
```

### adc\_group\_synchronous\_trigger\_enable

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

**Table 3-70. Function adc\_group\_synchronous\_trigger\_enable**

<b>Function name</b>	adc_group_synchronous_trigger_enable
<b>Function prototype</b>	void adc_group_synchronous_trigger_enable(uint32_t adc_periph, adc_group_select_enum group)
<b>Function descriptions</b>	enable ADC synchronous trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum</a>



	<a href="#"><u>adc_group_select_enum.</u></a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC synchronous trigger */
```

```
adc_group_synchronous_trigger_enable (ADC0, ADC_GROUP_PRI2);
```

### adc\_group\_synchronous\_trigger\_source\_config

The description of adc\_group\_synchronous\_trigger\_source\_config is shown as below:

**Table 3-71. Function adc\_group\_synchronous\_trigger\_source\_config**

<b>Function name</b>	adc_group_synchronous_trigger_source_config
<b>Function prototype</b>	void adc_group_synchronous_trigger_source_config(uint32_t adc_periph, adc_group_select_enum group, uint32_t external_trigger_source)
<b>Function descriptions</b>	configure ADC synchronous trigger source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#"><u>Enum adc_group_select_enum.</u></a>
<b>Input parameter{in}</b>	
<b>external_trigger_source</b>	trigger source
<i>ADC_SYNCTRIG_SOURCE_TIMER0_TRGOF</i>	TIMER0_TRGOF event select
<i>ADC_SYNCTRIG_SOURCE_TIMER0_TRGUF</i>	TIMER0_TRGUF event select
<i>ADC_SYNCTRIG_SOURCE_TIMER0_TRGA</i>	TIMER0_TRGA event select
<i>ADC_SYNCTRIG_SOURCE_TIMER0_TRGB</i>	TIMER0_TRGB event select
<i>ADC_SYNCTRIG_SOURCE_TIMER0_TRGAB</i>	TIMER0_TRGAB event select
<i>ADC_SYNCTRIG_SOURCE</i>	TIMER0_TRGAORB event select (for ADC extend bifurcate trigger source)

URCE_TIMER0_TRGA ORB	
ADC_SYNCTRIG_SO URCE_TIMER0_TRG O	TIMER0_TRGO event select
ADC_SYNCTRIG_SO URCE_TIMER7_TRG OF	TIMER7_TRGOF event select
ADC_SYNCTRIG_SO URCE_TIMER7_TRGU F	TIMER7_TRGUF event select
ADC_SYNCTRIG_SO URCE_TIMER7_TRGA	TIMER7_TRGA event select
ADC_SYNCTRIG_SO URCE_TIMER7_TRGB	TIMER7_TRGB event select
ADC_SYNCTRIG_SO URCE_TIMER7_TRGA B	TIMER7_TRGAB event select
ADC_SYNCTRIG_SO URCE_TIMER7_TRGA ORB	TIMER7_TRGAORB event select (for ADC extend bifurcate trigger source)
ADC_SYNCTRIG_SO URCE_TIMER7_TRG O	TIMER7_TRGO event select
ADC_SYNCTRIG_SO URCE_TIMER1_TRG O	TIMER1_TRGO event select
ADC_SYNCTRIG_SO URCE_TIMER2_TRG O	TIMER2_TRGO event select
ADC_SYNCTRIG_SO URCE_GPTIMER0_TR GA	GPTIMER0_TRGA event select
ADC_SYNCTRIG_SO URCE_GPTIMER0_TR GB	GPTIMER0_TRGB event select
ADC_SYNCTRIG_SO URCE_GPTIMER0_TR GAB	GPTIMER0_TRGAB event select (for ADC extend bifurcate trigger source)
ADC_SYNCTRIG_SO URCE_GPTIMER1_TR GA	GPTIMER1_TRGA event select
ADC_SYNCTRIG_SO URCE_GPTIMER1_TR	GPTIMER1_TRGB event select

<i>GB</i>	
<i>ADC_SYNCTRIG_SO</i> <i>URCE_GPTIMER1_TR</i> <i>GAB</i>	GPTIMER1_TRGAB event select (for ADC extend bifurcate trigger source)
<i>ADC_SYNCTRIG_SO</i> <i>URCE_EVIC_EVSEL0</i>	EVIC_EVSEL0 event select
<i>ADC_SYNCTRIG_SO</i> <i>URCE_EVIC_EVSEL1</i>	EVIC_EVSEL1 event select
<i>ADC_SYNCTRIG_SO</i> <i>URCE_EVIC_EVSEL0</i> <i>_OR_1</i>	EVIC_EVSEL0 or EVIC_EVSEL1 event select
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC synchronous trigger source */
```

```
adc_group_synchronous_trigger_source_config      (ADC0,      ADC_GROUP_PRI2,  
ADC_SYNCTRIG_SOURCE_TIMER0_TRGB);
```

### adc\_group\_asynchronous\_trigger\_enable

The description of `adc_group_asynchronous_trigger_enable` is shown as below:

**Table 3-72. Function `adc_group_asynchronous_trigger_enable`**

<b>Function name</b>	<code>adc_group_asynchronous_trigger_enable</code>
<b>Function prototype</b>	<code>void adc_group_asynchronous_trigger_enable(uint32_t adc_periph, adc_group_select_enum group)</code>
<b>Function descriptions</b>	enable ADC asynchronous trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum <code>adc_group_select_enum</code></a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ADC asynchronous trigger */
```

```
adc_group_asynchronous_trigger_enable (ADC0, ADC_GROUP_PRI2);
```

### adc\_group\_software\_trigger\_enable

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

**Table 3-73. Function adc\_group\_software\_trigger\_enable**

<b>Function name</b>	adc_group_software_trigger_enable
<b>Function prototype</b>	void adc_group_software_trigger_enable(uint32_t adc_periph, adc_group_select_enum group)
<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
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ADC software trigger */
```

```
adc_group_software_trigger_enable (ADC0, ADC_GROUP_PRI2);
```

### adc\_group\_software\_end\_conversion

The description of adc\_group\_software\_end\_conversion is shown as below:

**Table 3-74. Function adc\_group\_software\_end\_conversion**

<b>Function name</b>	adc_group_software_end_conversion
<b>Function prototype</b>	void adc_group_software_end_conversion(uint32_t adc_periph)
<b>Function descriptions</b>	software end conversion of all groups
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* software end conversion of all groups */
```

```
adc_group_software_end_conversion (ADC0);
```

### adc\_group\_bifurcate\_mode\_enable

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

**Table 3-75. Function adc\_group\_bifurcate\_mode\_enable**

Function name	adc_group_bifurcate_mode_enable
Function prototype	void adc_group_bifurcate_mode_enable(uint32_t adc_periph, adc_group_select_enum group)
Function descriptions	enable ADC bifurcate trigger mode for Group_pri x
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
Input parameter{in}	
group	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC bifurcate trigger mode for Group_pri x */
```

```
adc_group_bifurcate_mode_enable (ADC0, ADC_GROUP_PRI1);
```

### adc\_group\_bifurcate\_mode\_disable

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

**Table 3-76. Function adc\_group\_bifurcate\_mode\_disable**

Function name	adc_group_bifurcate_mode_disable
Function prototype	void adc_group_bifurcate_mode_disable(uint32_t adc_periph, adc_group_select_enum group)
Function descriptions	disable ADC bifurcate trigger mode for Group_pri x
Precondition	-
The called functions	-

Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
Input parameter{in}	
<b>group</b>	select the group. The members can refer to <a href="#">Enum <i>adc_group_select_enum</i></a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable ADC bifurcate trigger mode for Group_pri x */
```

```
adc_group_bifurcate_mode_disable (ADC0, ADC_GROUP_PRI1);
```

### adc\_group\_bifurcate\_channel\_select

The description of `adc_group_bifurcate_channel_select` is shown as below:

**Table 3-77. Function `adc_group_bifurcate_channel_select`**

<b>Function name</b>	<code>adc_group_bifurcate_channel_select</code>
<b>Function prototype</b>	<code>void adc_group_bifurcate_channel_select(uint32_t adc_periph, adc_group_select_enum group, adc_channel_select_enum channel)</code>
<b>Function descriptions</b>	ADC bifurcate trigger channel select
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
Input parameter{in}	
<b>group</b>	select the group. The members can refer to <a href="#">Enum <i>adc_group_select_enum</i></a> .
Input parameter{in}	
<b>channel</b>	bifurcate trigger mode channel select, refer to <a href="#">Enum <i>adc_watchdog_compare_condition_enum</i></a> . <b>Note:</b> except for <code>ADC_CHANNEL_ALL</code> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* ADC bifurcate trigger channel select */
```

```
adc_group_bifurcate_channel_select      (ADC0,      ADC_GROUP_PRI1,
ADC_BICHSEL_CHANNEL_IN00);
```

### adc\_group\_bifurcate\_extend\_trigger\_select

The description of adc\_group\_bifurcate\_extend\_trigger\_select is shown as below:

**Table 3-78. Function adc\_group\_bifurcate\_extend\_trigger\_select**

<b>Function name</b>	adc_group_bifurcate_extend_trigger_select
<b>Function prototype</b>	void adc_group_bifurcate_extend_trigger_select(uint32_t adc_periph, adc_group_select_enum group, uint32_t trigger_source)
<b>Function descriptions</b>	extend bifurcate trigger source select
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> .
<b>Input parameter{in}</b>	
<b>trigger_source</b>	extend double trigger source select
<i>ADC_SYNCTRIG_SOURCE_TIMER0_TRGAORB</i>	select TIMER0_TRGAORB as double trigger source for ADCx(x=0,2)
<i>ADC_SYNCTRIG_SOURCE_TIMER7_TRGAORB</i>	select TIMER7_TRGAORB as double trigger source for ADCx(x=0,2)
<i>ADC_SYNCTRIG_SOURCE_GPTIMER0_TRGAB</i>	select GPTIMER0_TRGAB as double trigger source for ADCx(x=0,2)
<i>ADC_SYNCTRIG_SOURCE_GPTIMER1_TRGAB</i>	select GPTIMER1_TRGAB as double trigger source for ADCx(x=0,2)
<i>ADC_SYNCTRIG_SOURCE_EVIC_EVSEL0_OR_1</i>	select EVIC_EVSEL0 or EVIC_EVSEL1 for ADC trigger source for ADCx(x=0,2)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* extend bifurcate trigger source select */
```

```
adc_group_bifurcate_extend_trigger_select (ADC0, ADC_GROUP_PRI1,
ADC_SYNCTRIG_SOURCE_TIMER0_TRGAORB);
```

### adc\_group\_bifurcate\_trigger\_restart\_enable

The description of adc\_group\_bifurcate\_trigger\_restart\_enable is shown as below:

**Table 3-79. Function adc\_group\_bifurcate\_trigger\_restart\_enable**

<b>Function name</b>	adc_group_bifurcate_trigger_restart_enable
<b>Function prototype</b>	void adc_group_bifurcate_trigger_restart_enable(uint32_t adc_periph, adc_group_select_enum group)
<b>Function descriptions</b>	enable restore of the next trigger during the current A/D conversion round
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable restore of the next trigger during the current A/D conversion round */
```

```
adc_group_bifurcate_trigger_restart_enable (ADC0, ADC_GROUP_PRI1);
```

### adc\_group\_bifurcate\_trigger\_restart\_disable

The description of adc\_group\_bifurcate\_trigger\_restart\_disable is shown as below:

**Table 3-80. Function adc\_group\_bifurcate\_trigger\_restart\_disable**

<b>Function name</b>	adc_group_bifurcate_trigger_restart_disable
<b>Function prototype</b>	void adc_group_bifurcate_trigger_restart_disable(uint32_t adc_periph, adc_group_select_enum group)
<b>Function descriptions</b>	disable restore of the next trigger during the current A/D conversion round
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum</a>



	<a href="#"><u>adc_group_select_enum.</u></a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable restore of the next trigger during the current A/D conversion round */
```

```
adc_group_bifurcate_trigger_restart_disable (ADC0, ADC_GROUP_PRI1);
```

### adc\_group\_dma\_mode\_enable

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

**Table 3-81. Function adc\_group\_dma\_mode\_enable**

Function name	adc_group_dma_mode_enable
Function prototype	void adc_group_dma_mode_enable(uint32_t adc_periph, adc_group_select_enum group)
Function descriptions	enable DMA request
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
Input parameter{in}	
group	select the group. The members can refer to <a href="#"><u>Enum</u></a> <a href="#"><u>adc_group_select_enum.</u></a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
adc_group_dma_mode_enable (ADC0, ADC_GROUP_PRI1);
```

### adc\_group\_dma\_mode\_disable

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

**Table 3-82. Function adc\_group\_dma\_mode\_disable**

Function name	adc_group_dma_mode_disable
Function prototype	void adc_group_dma_mode_disable(uint32_t adc_periph, adc_group_select_enum group)

<b>Function descriptions</b>	disable DMA request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum <i>adc_group_select_enum</i></a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMA request */

adc_group_dma_mode_disable (ADC0, ADC_GROUP_PRI1);
```

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

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

**Table 3-83. Function adc\_group\_dma\_request\_after\_last\_enable**

<b>Function name</b>	adc_group_dma_request_after_last_enable
<b>Function prototype</b>	void adc_group_dma_request_after_last_enable(uint32_t adc_periph, adc_group_select_enum group)
<b>Function descriptions</b>	when DMAENx=1, the DMA engine issues request at end of conversion of Group_prix
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum <i>adc_group_select_enum</i></a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* when DMAENx=1, the DMA engine issues request at end of conversion of Group_prix */
```

adc\_group\_dma\_request\_after\_last\_enable (ADC0, ADC\_GROUP\_PRI1);

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

The description of adc\_group\_dma\_request\_after\_last\_disable is shown as below:

**Table 3-84. Function adc\_group\_dma\_request\_after\_last\_disable**

<b>Function name</b>	adc_group_dma_request_after_last_disable
<b>Function prototype</b>	void adc_group_dma_request_after_last_disable(uint32_t adc_periph, adc_group_select_enum group)
<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=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> .
<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
*/
```

adc\_group\_dma\_request\_after\_last\_disable (ADC0, ADC\_GROUP\_PRI1);

### adc\_group\_dma\_overshoot\_detect\_enable

The description of adc\_group\_dma\_overshoot\_detect\_enable is shown as below:

**Table 3-85. Function adc\_group\_dma\_overshoot\_detect\_enable**

<b>Function name</b>	adc_group_dma_overshoot_detect_enable
<b>Function prototype</b>	void adc_group_dma_overshoot_detect_enable(uint32_t adc_periph, adc_group_select_enum group)
<b>Function descriptions</b>	enable dma overshoot detect
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	

group	select the group. The members can refer to <a href="#">Enum</a> <a href="#">adc_group_select_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable dma overrun detect */
```

```
adc_group_dma_overrun_detect_enable (ADC0, ADC_GROUP_PRI1);
```

### adc\_group\_dma\_overrun\_detect\_disable

The description of adc\_group\_dma\_overrun\_detect\_disable is shown as below:

**Table 3-86. Function adc\_group\_dma\_overrun\_detect\_disable**

Function name	adc_group_dma_overrun_detect_disable
Function prototype	void adc_group_dma_overrun_detect_disable(uint32_t adc_periph, adc_group_select_enum group)
Function descriptions	disable dma overrun detect
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
Input parameter{in}	
group	select the group. The members can refer to <a href="#">Enum</a> <a href="#">adc_group_select_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable dma overrun detect */
```

```
adc_group_dma_overrun_detect_disable (ADC0, ADC_GROUP_PRI1);
```

### adc\_channel\_priority\_config

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

**Table 3-87. Function adc\_channel\_priority\_config**

Function name	adc_channel_priority_config
Function prototype	void adc_channel_priority_config(uint32_t adc_periph,

	adc_channel_select_enum adc_channel, uint8_t rank)
<b>Function descriptions</b>	channel priority config
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	bifurcate trigger mode channel select. The members can refer to <a href="#">Enum adc_watchdog_compare_condition_enum</a> . <b>Note:</b> except for <i>ADC_CHANNEL_ALL</i> .
<b>Input parameter{in}</b>	
<b>rank</b>	the priority of the selected channel
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* channel priority config */
```

```
adc_channel_priority_config (ADC0, ADC_CHANNEL_IN01, 2);
```

```
adc_channel_priority_config (ADC0, ADC_CHANNEL_IN02, 1);
```

### adc\_channel\_sample\_time\_config

The description of `adc_channel_sample_time_config` is shown as below:

**Table 3-88. Function `adc_channel_sample_time_config`**

<b>Function name</b>	adc_channel_sample_time_config
<b>Function prototype</b>	void adc_channel_sample_time_config(uint32_t adc_periph, adc_channel_select_enum channel, uint32_t sample_time)
<b>Function descriptions</b>	configure ADC channel sample time
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	bifurcate trigger mode channel select. The members can refer to <a href="#">Enum adc_watchdog_compare_condition_enum</a> .
<b>Input parameter{in}</b>	
<b>sample_time</b>	0x02~0xFF. The sample time value 0x02 cycle.
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure ADC channel sample time */
```

```
adc_channel_sample_time_config (ADC0, ADC_CHANNEL_IN01, 0x02);
```

```
adc_channel_sample_time_config (ADC0, ADC_CHANNEL_IN02, 0x02);
```

### adc\_evic\_link\_signal\_event\_config

The description of adc\_evic\_link\_signal\_event\_config is shown as below:

**Table 3-89. Function adc\_evic\_link\_signal\_event\_config**

<b>Function name</b>	adc_evic_link_signal_event_config
<b>Function prototype</b>	void adc_evic_link_signal_event_config(uint32_t adc_periph, uint32_t linksignal)
<b>Function descriptions</b>	select which event as the EVIC link signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>linksignal</b>	EVIC link signal
ADC_EVIC_LINK_GRP_PRI1	an event signal is generated when scanning for Group_pri1 is completed
ADC_EVIC_LINK_GRP_PRI2	an event signal is generated when scanning for Group_pri2 is completed
ADC_EVIC_LINK_GRP_PRI3	an event signal is generated when scanning for Group_pri3 is completed
ADC_EVIC_LINK_GRP_PRI4	an event signal is generated when scanning for Group_pri4 is completed
ADC_EVIC_LINK_GRP_ALL	an event signal is generated when scanning for Group_pri1, Group_pri2, Group_pri3 or Group_pri4 is completed
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select which event as the EVIC link signal */
```

```
adc_evic_link_signal_event_config (ADC0, ADC_EVIC_LINK_GROUP_PRI2);
```

## adc\_group\_evic\_link\_signal\_source

The description of adc\_group\_evic\_link\_signal\_source is shown as below:

**Table 3-90. Function adc\_group\_evic\_link\_signal\_source**

<b>Function name</b>	adc_group_evic_link_signal_source
<b>Function prototype</b>	void adc_group_evic_link_signal_source(uint32_t adc_periph, adc_group_select_enum group, adc_evic_link_source_enum signal_src)
<b>Function descriptions</b>	select EVIC link signal source for Group pri x
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>group</b>	select the group. The members can refer to <a href="#">Enum adc_group_select_enum</a> .
<b>Input parameter{in}</b>	
<b>signal_src</b>	EVIC trigger signal source. The members can refer to <a href="#">Enum adc_evic_link_source_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select which event as the EVIC link signal */
```

```
adc_evic_link_signal_event_config      (ADC0,          ADC_GROUP_PRI1,  
ADC_EVIC_LINK_SOURCE_EOCR_FFLAG);
```

## adc\_channel\_data\_read

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

**Table 3-91. Function adc\_channel\_data\_read**

<b>Function name</b>	adc_channel_data_read
<b>Function prototype</b>	uint16_t adc_channel_data_read(uint32_t adc_periph, adc_channel_select_enum channel)
<b>Function descriptions</b>	read ADC channel data register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection

Input parameter{in}	
channel	the selected ADC channel. The members can refer <a href="#">Enum adc_watchdog_compare_condition_enum</a> . <b>Note:</b> except for ADC_CHANNEL_ALL.
Output parameter{out}	
-	-
Return value	
the conversion value	0~0xFFFF

Example:

```
/* read ADC channel data register */
```

```
uint16_t channel1_data;
```

```
channel1_data = adc_channel_data_read (ADC0, ADC_CHANNEL_IN01);
```

### adc\_self\_diagnosis\_data\_read

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

**Table 3-92. Function adc\_self\_diagnosis\_data\_read**

<b>Function name</b>	adc_self_diagnosis_data_read
<b>Function prototype</b>	uint16_t adc_self_diagnosis_data_read(uint32_t adc_periph)
<b>Function descriptions</b>	read ADC self-diagnosis data
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
the conversion value	0x0000~0x0FFF (DAL=0,LSB alignment), 0x0000~0xFFFF (DAL=1,MSB alignment),

Example:

```
/* read ADC self-diagnosis data */
```

```
uint16_t data;
```

```
data = adc_self_diagnosis_data_read (ADC0);
```

### adc\_self\_diagnosis\_status\_read

The description of adc\_self\_diagnosis\_status\_read is shown as below:



Table 3-93. Function `adc_self_diagnosis_status_read`

<b>Function name</b>	<code>adc_self_diagnosis_status_read</code>
<b>Function prototype</b>	<code>uint16_t adc_self_diagnosis_status_read(uint32_t adc_periph)</code>
<b>Function descriptions</b>	read ADC self-diagnosis status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<code>ADCx(x=0,2)</code>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>voltage status converted by self-diagnosis</b>	0x0: self-diagnosis has never been executed since power-on. 0x1: 0V converted by self-diagnosis 0x2: $V_{REFP}/2$ converted by self-diagnosis 0x3: $V_{REFP}$ converted by self-diagnosis

Example:

```

/* read ADC self-diagnosis status */

uint16_t status;

status= adc_self_diagnosis_status_read (ADC0);

```

### `adc_bifurcate_data_read`

The description of `adc_bifurcate_data_read` is shown as below:

Table 3-94. Function `adc_bifurcate_data_read`

<b>Function name</b>	<code>adc_bifurcate_data_read</code>
<b>Function prototype</b>	<code>uint16_t adc_bifurcate_data_read(uint32_t adc_periph, adc_bifurcate_data_enum channel)</code>
<b>Function descriptions</b>	read ADC bifurcate data register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<code>ADCx(x=0,2)</code>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	data channel select. The members can refer to <a href="#"><u>Enum adc_bifurcate_data_enum</u></a> .
<code>ADC_BIFURCATE_DATA</code>	read duplication data select
<code>ADC_EXTENDEDED_BIFURCATE_DATA_1</code>	read duplication 1 data

ADC_EXTENDED_BIFURCATE_DATA_2	read duplication 2 data
Output parameter{out}	
-	-
Return value	
the conversion value	0~0xFFFF

Example:

```
/* read ADC bifurcate data register */

uint16_t data;

data = adc_bifurcate_data_read (ADC0, ADC_BIFURCATE_DATA);
```

### adc\_flag\_get

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

**Table 3-95. Function adc\_flag\_get**

Function name	adc_flag_get
Function prototype	FlagStatus adc_flag_get(uint32_t adc_periph, adc_event_flag_enum flag)
Function descriptions	get the ADC flag
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,2)	ADC peripheral selection
Input parameter{in}	
flag	the ADC flag. The members can refer to <a href="#">Table 3-16. Enum adc_event_flag_enum</a> .
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the ADC flag */

uint16_t flag;

flag = adc_flag_get (ADC0, ADC_FLAG_EOC1RF);
```

### adc\_flag\_clear

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

Table 3-96. Function `adc_flag_clear`

Function name	<code>adc_flag_clear</code>
Function prototype	<code>void adc_flag_clear(uint32_t adc_periph, adc_event_flag_enum flag)</code>
Function descriptions	clear the ADC flag
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_periph</code>	ADC peripheral
<code>ADCx(x=0,2)</code>	ADC peripheral selection
Input parameter{in}	
<code>flag</code>	the ADC flag. The members can refer to <a href="#">Table 3-16. Enum <code>adc_event_flag_enum</code></a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the ADC flag */
```

```
adc_flag_clear (ADC0, ADC_FLAG_EOC1RF);
```

### `adc_interrupt_enable`

The description of `adc_interrupt_enable` is shown as below:

Table 3-97. Function `adc_interrupt_enable`

Function name	<code>adc_interrupt_enable</code>
Function prototype	<code>void adc_interrupt_enable(uint32_t adc_periph, adc_interrupt_enum interrupt)</code>
Function descriptions	enable ADC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_periph</code>	ADC peripheral
<code>ADCx(x=0,2)</code>	ADC peripheral selection
Input parameter{in}	
<code>flag</code>	the interrupt select. The members can refer to <a href="#">Enum <code>adc_interrupt_enum</code></a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ADC interrupt */
```

```
adc_interrupt_enable (ADC0, ADC_INT_EOC1RF);
```

### adc\_interrupt\_disable

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

**Table 3-98. Function adc\_interrupt\_disable**

<b>Function name</b>	adc_interrupt_disable
<b>Function prototype</b>	void adc_interrupt_disable(uint32_t adc_periph, adc_interrupt_enum interrupt)
<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=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>flag</b>	the interrupt select. The members can refer to <a href="#">Enum adc_interrupt_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ADC interrupt */
```

```
adc_interrupt_disable (ADC0, ADC_INT_EOC1RF);
```

### adc\_interrupt\_flag\_get

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

**Table 3-99. Function adc\_interrupt\_flag\_get**

<b>Function name</b>	adc_interrupt_flag_get
<b>Function prototype</b>	FlagStatus adc_interrupt_flag_get(uint32_t adc_periph, adc_interrupt_flag_enum int_flag)
<b>Function descriptions</b>	get ADC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	

<b>int_flag</b>	the ADC interrupt flag. The members can refer to <a href="#">Enum <i>adc_interrupt_flag_enum</i></a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus: SET or RESET</b>	

Example:

```
/* get ADC interrupt flag */
```

```
uint16_t flag;
```

```
flag = adc_interrupt_flag_get (ADC0, ADC_INT_FLAG_EOC1RF);
```

### adc\_interrupt\_flag\_clear

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

**Table 3-100. 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, adc_interrupt_flag_enum int_flag)
<b>Function descriptions</b>	clear ADC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>int_flag</b>	the ADC interrupt flag. The members can refer to <a href="#">Enum <i>adc_interrupt_flag_enum</i></a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear ADC interrupt flag */
```

```
adc_interrupt_flag_clear (ADC0, ADC_INT_FLAG_EOC1RF);
```

## 3.3. CAN

CAN bus (for Controller Area Network) is a bus standard designed to allow microcontrollers

and devices to communicate with each other without a host computer. 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 3-101. CAN Registers**

Registers	Descriptions
CAN_CTL	Control register
CAN_STAT	Status register
CAN_TSTAT	Transmit status register
CAN_RFIFO0	Receive message FIFO0 register
CAN_RFIFO1	Receive message FIFO1 register
CAN_INTEN	Interrupt enable register
CAN_ERR	Error register
CAN_BT	Bit timing register
CAN_TMIx	Transmit mailbox identifier register
CAN_TMPx	Transmit mailbox property register
CAN_TMDATA0x	Transmit mailbox data0 register
CAN_TMDATA1x	Transmit mailbox data1 register
CAN_RFIFOMIx	Receive FIFO mailbox identifier register
CAN_RFIFOMPx	Receive FIFO mailbox property register
CAN_RFIFOMDAT A0x	Receive FIFO mailbox data0 register
CAN_RFIFOMDAT A1x	Receive FIFO mailbox data1 register
CAN_FCTL	Filter control register
CAN_FMCFG	Filter mode configuration register
CAN_FSCFG	Filter scale configuration register
CAN_FAFIFO	Filter associated FIFO registe
CAN_FW	Filter working register
CAN_FxDATAy	Filter x data y register

### 3.3.2. Descriptions of Peripheral functions

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

**Table 3-102. CAN firmware function**

Function name	Function description
can_deinit	deinitialize CAN
can_struct_para_init	initialize CAN parameter struct with a default value
can_init	initialize CAN
can_filter_init	initialize CAN filter

Function name	Function description
can_debug_freeze_enable	CAN debug freeze enable
can_debug_freeze_disable	CAN debug freeze disable
can_time_trigger_mode_enable	CAN time trigger mode enable
can_time_trigger_mode_disable	CAN time trigger mode disable
can_message_transmit	transmit CAN message
can_transmit_states	get CAN transmit state
can_transmission_stop	stop CAN transmission
can_message_receive	CAN receive message
can_fifo_release	CAN release fifo
can_receive_message_length_get	CAN receive message length
can_working_mode_set	CAN working mode
can_wakeup	CAN wakeup from sleep mode
can_error_get	CAN get error
can_receive_error_number_get	get CAN receive error number
can_transmit_error_number_get	get CAN transmit error number
can_flag_get	CAN get flag state
can_flag_clear	CAN clear flag state
can_interrupt_enable	CAN interrupt enable
can_interrupt_disable	CAN interrupt disable
can_interrupt_flag_get	CAN get interrupt flag state
can_interrupt_flag_clear	CAN clear interrupt flag state

### Structure can\_parameter\_struct

**Table 3-103. Structure can\_parameter\_struct**

Member name	Function description
working_mode	CAN working mode
resync_jump_width	CAN resynchronization jump width
time_segment_1	time segment 1
time_segment_2	time segment 2
bit_sample_mode	bit sampling synchronization select
sample_bit_select	sample bit select
frame_inter_to_id	frame inter to identifier enable
time_triggered	time triggered communication mode
auto_bus_off_recovery	automatic bus-off recovery
auto_wake_up	automatic wake-up mode
auto_retrans	automatic retransmission mode
rec_fifo_overwrite	receive FIFO overwrite mode
trans_fifo_order	transmit FIFO order
prescaler	baudrate prescaler

## Structure can\_trasnmit\_message\_struct

**Table 3-104. Structure can\_trasnmit\_message\_struct**

Member name	Function description
tx_sfids	standard format frame identifier
tx_efids	extended format frame identifier
tx_ff	format of frame: standard or extended format
tx_ft	type of frame: data or remote
tx_dlen	data length
reserved	reserved byte for alignment
tx_data[8]	transmit data

## Structure can\_receive\_message\_struct

**Table 3-105. Structure can\_receive\_message\_struct**

Member name	Function description
rx_sfids	standard format frame identifier
rx_efids	extended format frame identifier
rx_ff	format of frame: standard or extended format
rx_ft	type of frame: data or remote
rx_dlen	data length
rx_fi	filtering index
rx_data[8]	receive data

## Structure can\_filter\_parameter\_struct

**Table 3-106. Structure can\_filter\_parameter\_struct**

Member name	Function description
filter_list_high	filter list number high bits
filter_list_low	filter list number low bits
filter_mask_high	filter mask number high bits
filter_mask_low	filter mask number low bits
filter_fifo_number	receive FIFO associated with the filter
filter_number	filter number
filter_mode	filter mode: list or mask
filter_bits	filter scale
filter_enable	filter work or not

## can\_deinit

The description of can\_deinit is shown as below:

**Table 3-107. Function can\_deinit**

Function name	can_deinit
Function prototype	void can_deinit(void);



<b>Function descriptions</b>	deinitialize CAN
<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:

```
/* CAN deinitialize*/
```

```
can_deinit ();
```

### can\_struct\_para\_init

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

**Table 3-108. 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 struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>type</b>	Sturct type
CAN_INIT_STRUCT	CAN initilaze parameters struct
CAN_FILTER_STRUCT	CAN filter parameters struct
CAN_TX_MESSAGE_STRUCT	CAN transmit message struct
CAN_RX_MESSAGE_STRUCT	CAN receive message struct
<b>Output parameter{out}</b>	
<b>p_struct</b>	the struct pointer that needs initialize
<b>Return value</b>	
-	-

Example:

```
can_parameter_struct can_init;
```

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

### can\_init

The description of can\_init is shown as below:

Table 3-109. Function can\_init

Function name	can_init
Function prototype	ErrStatus can_init(can_parameter_struct* can_parameter_init);
Function descriptions	initialize CAN
Precondition	can_struct_para_init()
The called functions	-
Input parameter{in}	
can_parameter_init	CAN parameter initialization struct, the structure members can refer to members of the structure <a href="#">Table 3-103. Structure can_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
ErrStatus	SUCCESS / ERROR

Example:

```
can_parameter_struct      can_parameter;

/* CAN initialize*/

can_init (&can_parameter);
```

### can\_filter\_init

The description of can\_filter\_init is shown as below:

Table 3-110. Function can\_filter\_init

Function name	can_filter_init
Function prototype	void can_filter_init(can_filter_parameter_struct* can_filter_parameter_init);
Function descriptions	initialize CAN filter
Precondition	can_struct_para_init()
The called functions	-
Input parameter{in}	
can_filter_parameter_init	CAN filter initialization struct, the structure members can refer to members of the structure <a href="#">Table 3-106. Structure can_filter_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize CAN filter */

can_filter_init(&can_filter);
```

## can\_debug\_freeze\_enable

The description of can\_debug\_freeze\_enable is shown as below:

**Table 3-111. Function can\_debug\_freeze\_enable**

<b>Function name</b>	can_debug_freeze_enable
<b>Function prototype</b>	void can_debug_freeze_enable(void);
<b>Function descriptions</b>	enable CAN debug freeze
<b>Precondition</b>	-
<b>The called functions</b>	dbg_periph_enable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CAN debug freeze */
can_debug_freeze_enable ();
```

## can\_debug\_freeze\_disable

The description of can\_debug\_freeze\_disable is shown as below:

**Table 3-112. Function can\_debug\_freeze\_disable**

<b>Function name</b>	can_debug_freeze_disable
<b>Function prototype</b>	void can_debug_freeze_disable(void);
<b>Function descriptions</b>	disable CAN debug freeze
<b>Precondition</b>	-
<b>The called functions</b>	dbg_periph_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CAN debug freeze */
can_debug_freeze_disable ();
```

## can\_time\_trigger\_mode\_enable

The description of can\_time\_trigger\_mode\_enable is shown as below:

**Table 3-113. Function can\_time\_trigger\_mode\_enable**

<b>Function name</b>	can_time_trigger_mode_enable
<b>Function prototype</b>	void can_time_trigger_mode_enable(void);
<b>Function descriptions</b>	enable CAN time trigger 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 CAN time trigger mode */
can_time_trigger_mode_enable ();
```

### can\_time\_trigger\_mode\_disable

The description of can\_time\_trigger\_mode\_disable is shown as below:

**Table 3-114. Function can\_time\_trigger\_mode\_disable**

<b>Function name</b>	can_time_trigger_mode_disable
<b>Function prototype</b>	void can_time_trigger_mode_disable(void);
<b>Function descriptions</b>	disable CAN time trigger 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 CAN time trigger mode */
can_time_trigger_mode_disable ();
```

### can\_message\_transmit

The description of can\_message\_transmit is shown as below:

**Table 3-115. Function can\_message\_transmit**

<b>Function name</b>	can_message_transmit
----------------------	----------------------

<b>Function prototype</b>	uint8_t can_message_transmit(can_transmit_message_struct* transmit_message);
<b>Function descriptions</b>	transmit CAN message
<b>Precondition</b>	can_struct_para_init()
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>transmit_message</b>	CAN transmit message struct, the structure members can refer to members of the structure <a href="#">Table 3-104. Structure can_transmit_message_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint8_t	0x00-0x03

Example:

```

/* CAN transmit message and return the mailbox number*/

uint8_t transmit_mailbox = 0;

transmit_mailbox = can_message_transmit(&transmit_message);

```

### can\_transmit\_states

The description of can\_transmit\_states is shown as below:

**Table 3-116. Function can\_transmit\_states**

<b>Function name</b>	can_transmit_states
<b>Function prototype</b>	can_transmit_state_enum can_transmit_states(uint8_t mailbox_number);
<b>Function descriptions</b>	get CAN transmit state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mailbox_number</b>	Mailbox number
CAN_MAILBOXx	CAN_MAILBOXx(x=0,1,2)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>can_transmit_state_enum</b>	0..4

Example:

```

/* CAN mailbox0 transmit state */

can_transmit_state_enum transmit_state = CAN_TRANSMIT_FAILED;

transmit_state = can_transmit_states (CAN_MAILBOX0);

```

## can\_transmission\_stop

The description of can\_transmission\_stop is shown as below:

**Table 3-117. Function can\_transmission\_stop**

<b>Function name</b>	can_transmission_stop
<b>Function prototype</b>	void can_transmission_stop(uint8_t mailbox_number);
<b>Function descriptions</b>	stop CAN transmission
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mailbox_number</b>	Mailbox number
<b>CAN_MAILBOXx</b>	CAN_MAILBOXx(x=0,1,2)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* stop CAN mailbox0 transmission */
can_transmission_stop (CAN_MAILBOX0);
```

## can\_message\_receive

The description of can\_message\_receive is shown as below:

**Table 3-118. Function can\_message\_receive**

<b>Function name</b>	can_message_receive
<b>Function prototype</b>	void can_message_receive(uint8_t fifo_number, can_receive_message_struct* receive_message);
<b>Function descriptions</b>	CAN receive message
<b>Precondition</b>	can_struct_para_init()
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fifo_number</b>	Fifo number
<b>CAN_FIFOx</b>	CAN_FIFOx(x=0,1)
<b>Input parameter{in}</b>	
<b>receive_message</b>	CAN message receive struct, the structure members can refer to members of the structure <a href="#">Table 3-105. Structure can_receive_message_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* CAN FIFO0 receive message */

can_message_receive(CAN_FIFO0, &receive_message);
```

### can\_fifo\_release

The description of can\_fifo\_release is shown as below:

**Table 3-119. Function can\_fifo\_release**

<b>Function name</b>	can_fifo_release
<b>Function prototype</b>	void can_fifo_release(uint8_t fifo_number);
<b>Function descriptions</b>	release FIFO0
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fifo_number</b>	Fifo number
<i>CAN_FIFOx</i>	CAN_FIFOx(x=0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* CAN release FIFO0*/

can_fifo_release (CAN_FIFO0);
```

### can\_receive\_message\_length\_get

The description of can\_receive\_message\_length\_get is shown as below:

**Table 3-120. Function can\_receive\_message\_length\_get**

<b>Function name</b>	can_receive_message_length_get
<b>Function prototype</b>	uint8_t can_receive_message_length_get(uint8_t fifo_number);
<b>Function descriptions</b>	CAN receive message length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fifo_number</b>	Fifo number
<i>CAN_FIFOx</i>	CAN_FIFOx(x=0,1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint8_t</b>	0..3

Example:

```
/* CAN FIFO0 receive message length */
uint8_t frame_number = 0;

frame_number = can_receive_message_length_get (CAN_FIFO0);
```

### can\_working\_mode\_set

The description of can\_working\_mode\_set is shown as below:

**Table 3-121. Function can\_working\_mode\_set**

<b>Function name</b>	can_working_mode_set
<b>Function prototype</b>	ErrStatus can_working_mode_set(uint8_t working_mode);
<b>Function descriptions</b>	set CAN working mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_working_mode</b>	Mode select
CAN_MODE_INITIALIZE	Initialize mode
CAN_MODE_NORMAL	Normal mode
CAN_MODE_SLEEP	Sleep mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS / ERROR

Example:

```
/* set CAN working at initialize mode */
can_working_mode_set (CAN_MODE_INITIALIZE);
```

### can\_wakeup

The description of can\_wakeup is shown as below:

**Table 3-122. Function can\_wakeup**

<b>Function name</b>	can_wakeup
<b>Function prototype</b>	ErrStatus can_wakeup(void);
<b>Function descriptions</b>	wake up CAN
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	



-	-
Return value	
ErrStatus	SUCCESS / ERROR

Example:

```
/* wake up CAN */
can_wakeup ();
```

### can\_error\_get

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

**Table 3-123. Function can\_error\_get**

Function name	can_error_get
Function prototype	can_error_enum can_error_get(void);
Function descriptions	get CAN error type
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
can_error_enum	0..7

Example:

```
/* get CAN error type */
can_error_get ();
```

### can\_receive\_error\_number\_get

The description of can\_receive\_error\_number\_get is shown as below:

**Table 3-124. Function can\_receive\_error\_number\_get**

Function name	can_receive_error_number_get
Function prototype	uint8_t can_receive_error_number_get(void);
Function descriptions	get CAN receive error number
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	

uint8_t	0..255
---------	--------

Example:

```
/* get CAN receive error number */

uint8_t error_num;

error_num = can_receive_error_number_get ();
```

### can\_transmit\_error\_number\_get

The description of can\_transmit\_error\_number\_get is shown as below:

**Table 3-125. Function can\_transmit\_error\_number\_get**

<b>Function name</b>	can_transmit_error_number_get
<b>Function prototype</b>	uint8_t can_transmit_error_number_get(void);
<b>Function descriptions</b>	get CAN transmit error number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint8_t	0..255

Example:

```
/* get CAN transmit error number */

uint8_t error_num;

error_num = can_transmit_error_number_get ();
```

### can\_flag\_get

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

**Table 3-126. Function can\_flag\_get**

<b>Function name</b>	can_flag_get
<b>Function prototype</b>	FlagStatus can_flag_get(an_flag_enum flag);
<b>Function descriptions</b>	get CAN flag state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
flag	CAN flags
CAN_FLAG_MTE2	mailbox 2 transmit error
CAN_FLAG_MTE1	mailbox 1 transmit error

CAN_FLAG_MTE0	mailbox 0 transmit error
CAN_FLAG_MTF2	mailbox 2 transmit finished
CAN_FLAG_MTF1	mailbox 1 transmit finished
CAN_FLAG_MTF0	mailbox 0 transmit finished
CAN_FLAG_RFO0	receive FIFO0 overfull
CAN_FLAG_RFF0	receive FIFO0 full
CAN_FLAG_RFO1	receive FIFO1 overfull
CAN_FLAG_RFF1	receive FIFO1 full
CAN_FLAG_BOERR	bus-off error
CAN_FLAG_PERR	passive error
CAN_FLAG_WERR	warning error
Output parameter{out}	
-	-
Return value	
FlagStatus	SET / RESET

Example:

```
/* get CAN mailbox 0 transmit finished flag */
```

```
FlagStatus can_flag = RESET;
```

```
can_flag = can_flag_get (CAN_FLAG_MTF0);
```

### can\_flag\_clear

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

**Table 3-127. Function can\_flag\_clear**

Function name	can_flag_clear
Function prototype	void can_flag_clear(can_flag_enum flag);
Function descriptions	clear CAN flag state
Precondition	-
The called functions	-
Input parameter{in}	
flag	CAN flags
CAN_FLAG_MTE2	mailbox 2 transmit error
CAN_FLAG_MTE1	mailbox 1 transmit error
CAN_FLAG_MTE0	mailbox 0 transmit error
CAN_FLAG_MTF2	mailbox 2 transmit finished
CAN_FLAG_MTF1	mailbox 1 transmit finished
CAN_FLAG_MTF0	mailbox 0 transmit finished
CAN_FLAG_RFO0	receive FIFO0 overfull
CAN_FLAG_RFF0	receive FIFO0 full
CAN_FLAG_RFO1	receive FIFO1 overfull
CAN_FLAG_RFF1	receive FIFO1 full

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear CAN mailbox 0 transmit error flag*/
```

```
can_flag_clear (CAN_FLAG_MTE0);
```

### can\_interrupt\_enable

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

**Table 3-128. Function can\_interrupt\_enable**

Function name	can_interrupt_enable
Function prototype	void can_interrupt_enable(uint32_t interrupt);
Function descriptions	enable CAN interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	Interrupt type
CAN_INT_TME	transmit mailbox empty interrupt enable
CAN_INT_RFNE0	receive FIFO0 not empty interrupt enable
CAN_INT_RFF0	receive FIFO0 full interrupt enable
CAN_INT_RFO0	receive FIFO0 overfull interrupt enable
CAN_INT_RFNE1	receive FIFO1 not empty interrupt enable
CAN_INT_RFF1	receive FIFO1 full interrupt enable
CAN_INT_RFO1	receive FIFO1 overfull interrupt enable
CAN_INT_WERR	warning error interrupt enable
CAN_INT_PERR	passive error interrupt enable
CAN_INT_BO	bus-off interrupt enable
CAN_INT_ERRN	error number interrupt enable
CAN_INT_ERR	error interrupt enable
CAN_INT_WU	wakeup interrupt enable
CAN_INT_SLPW	sleep working interrupt enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* CAN transmit mailbox empty interrupt enable */
```

```
can_interrupt_enable (CAN_INT_TME);
```

## can\_interrupt\_disable

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

**Table 3-129. Function can\_interrupt\_disable**

<b>Function name</b>	can_interrupt_disable
<b>Function prototype</b>	void can_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable CAN interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	Interrupt type
CAN_INT_TME	transmit mailbox empty interrupt enable
CAN_INT_RFNE0	receive FIFO0 not empty interrupt enable
CAN_INT_RFF0	receive FIFO0 full interrupt enable
CAN_INT_RFO0	receive FIFO0 overfull interrupt enable
CAN_INT_RFNE1	receive FIFO1 not empty interrupt enable
CAN_INT_RFF1	receive FIFO1 full interrupt enable
CAN_INT_RFO1	receive FIFO1 overfull interrupt enable
CAN_INT_WERR	warning error interrupt enable
CAN_INT_PERR	passive error interrupt enable
CAN_INT_BO	bus-off interrupt enable
CAN_INT_ERRN	error number interrupt enable
CAN_INT_ERR	error interrupt enable
CAN_INT_WU	wakeup interrupt enable
CAN_INT_SLPW	sleep working interrupt enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* CAN transmit mailbox empty interrupt disable */
can_interrupt_disable (CAN_INT_TME);
```

## can\_interrupt\_flag\_get

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

**Table 3-130. Function can\_interrupt\_flag\_get**

<b>Function name</b>	can_interrupt_flag_get
<b>Function prototype</b>	FlagStatus can_interrupt_flag_get(can_interrupt_flag_enum flag);
<b>Function descriptions</b>	get CAN interrupt flag state
<b>Precondition</b>	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	CAN interrupt flags
<i>CAN_INT_FLAG_SLPIF</i>	status change interrupt flag of sleep working mode entering
<i>CAN_INT_FLAG_WUIF</i>	status change interrupt flag of wakeup from sleep working mode
<i>CAN_INT_FLAG_ERRIF</i>	error interrupt flag
<i>CAN_INT_FLAG_MTF2</i>	mailbox 2 transmit finished interrupt flag
<i>CAN_INT_FLAG_MTF1</i>	mailbox 1 transmit finished interrupt flag
<i>CAN_INT_FLAG_MTF0</i>	mailbox 0 transmit finished interrupt flag
<i>CAN_INT_FLAG_RFO0</i>	receive FIFO0 overfull interrupt flag
<i>CAN_INT_FLAG_RFF0</i>	receive FIFO0 full interrupt flag
<i>CAN_INT_FLAG_RFO1</i>	receive FIFO1 overfull interrupt flag
<i>CAN_INT_FLAG_RFF1</i>	receive FIFO1 full interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET / RESET

Example:

```
/* get CAN mailbox 0 transmit finished interrupt flag */
```

```
FlagStatus can_int_flag = RESET;
```

```
can_flag = can_interrupt_flag_get (CAN_INT_FLAG_MTF0);
```

### can\_interrupt\_flag\_clear

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

**Table 3-131. Function can\_interrupt\_flag\_clear**

<b>Function name</b>	can_interrupt_flag_clear
<b>Function prototype</b>	void can_interrupt_flag_clear(can_interrupt_flag_enum flag);
<b>Function descriptions</b>	clear CAN interrupt flag state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1)</i>	CAN peripheral selection, the CAN1 only for GD32F30X_CL
<b>Input parameter{in}</b>	
<b>flag</b>	CAN interrupt flags
<i>CAN_INT_FLAG_SLPIF</i>	status change interrupt flag of sleep working mode entering
<i>CAN_INT_FLAG_WUIF</i>	status change interrupt flag of wakeup from sleep working mode

<code>CAN_INT_FLAG_ERRIF</code>	error interrupt flag
<code>CAN_INT_FLAG_MTF2</code>	mailbox 2 transmit finished interrupt flag
<code>CAN_INT_FLAG_MTF1</code>	mailbox 1 transmit finished interrupt flag
<code>CAN_INT_FLAG_MTF0</code>	mailbox 0 transmit finished interrupt flag
<code>CAN_INT_FLAG_RFO0</code>	receive FIFO0 overfull interrupt flag
<code>CAN_INT_FLAG_RFF0</code>	receive FIFO0 full interrupt flag
<code>CAN_INT_FLAG_RFO1</code>	receive FIFO1 overfull interrupt flag
<code>CAN_INT_FLAG_RFF1</code>	receive FIFO1 full interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear CAN mailbox 0 transmit finished interrupt flag */
can_interrupt_flag_clear (CAN_INT_FLAG_MTF0);
```

## 3.4. CFMU

The clock frequency measurement unit provides a clock frequency accuracy measurement function. The CFMU registers are listed in chapter [3.4.1](#), the CFMU firmware functions are introduced in chapter [3.4.2](#).

### 3.4.1. Descriptions of Peripheral registers

**Table 3-132. CFMU Registers**

Registers	Descriptions
CFMU_CTL	control register
CFMU_STAT	status register
CFMU_LVCFG	limit value configuration register
CFMU_CNT	counter value register

### 3.4.2. Descriptions of Peripheral functions

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

**Table 3-133. CFMU firmware function**

Function name	Function description
<code>cfmu_deinit</code>	deinitialize the CFMU
<code>cfmu_enable</code>	enable clock frequency measurement
<code>cfmu_disable</code>	disable clock frequency measurement

cfmu_cfmuref_enable	enable CFMUREF pin input
cfmu_cfmuref_disable	disable CFMUREF pin input
cfmu_reference_signal_config	configure the reference signal
cfmu_digital_filter_config	configure the digital filter
cfmu_reference_clock_config	configure the measurement reference clock source
cfmu_measurement_clock_config	configure the measurement target clock source
cfmu_limit_value_config	configure CFMU higher-limit value and lower-limit value
cfmu_interrupt_enable	enable the CFMU interrupt
cfmu_interrupt_disable	disable the CFMU interrupt
cfmu_interrupt_flag_get	get the CFMU interrupt flags
cfmu_interrupt_flag_clear	clear the CFMU interrupt flags

### Enum cfmu\_int\_enum

**Table 3-134. Enum cfmu\_int\_enum**

Member name	Descriptions
CFMU_INT_CFERR	clock frequency error interrupt request
CFMU_INT_CFMEND	clock frequency accuracy measurement end interrupt request
CFMU_INT_OVF	overflow interrupt request

### Enum cfmu\_int\_flag\_enum

**Table 3-135. Enum cfmu\_int\_flag\_enum**

Member name	Descriptions
CFMU_INT_FLAG_OVF	overflow flag
CFMU_INT_FLAG_CFMEND	clock frequency accuracy measurement end flag
CFMU_INT_FLAG_CFERR	clock frequency error flag

### Enum cfmu\_int\_flag\_clear\_enum

**Table 3-136. Enum cfmu\_int\_flag\_clear\_enum**

Member name	Descriptions
CFMU_INT_FLAG_CFERR_CLR	clock frequency error flag clear
CFMU_INT_FLAG_CFMEND_CLR	clock frequency accuracy measurement end flag clear
CFMU_INT_FLAG_OVF_CLR	overflow flag clear



F_CLR	
-------	--

### cfmu\_deinit

The description of cfmu\_deinit is shown as below:

**Table 3-137. Function cfmu\_deinit**

<b>Function name</b>	cfmu_deinit
<b>Function prototype</b>	void cfmu_deinit(void)
<b>Function descriptions</b>	deinitialize the CFMU
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize the CFMU */
cfmu_deinit();
```

### cfmu\_enable

The description of cfmu\_enable is shown as below:

**Table 3-138. Function cfmu\_enable**

<b>Function name</b>	cfmu_enable
<b>Function prototype</b>	void cfmu_enable(void)
<b>Function descriptions</b>	enable clock frequency measurement
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable clock frequency measurement */
cfmu_enable();
```

## cfmu\_disable

The description of cfmu\_disable is shown as below:

**Table 3-139. Function cfmu\_disable**

<b>Function name</b>	cfmu_disable
<b>Function prototype</b>	void cfmu_disable(void)
<b>Function descriptions</b>	disable clock frequency measurement
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable clock frequency measurement */
cfmu_disable();
```

## cfmu\_cfmuref\_enable

The description of cfmu\_cfmuref\_enable is shown as below:

**Table 3-140. Function cfmu\_cfmuref\_enable**

<b>Function name</b>	cfmu_cfmuref_enable
<b>Function prototype</b>	void cfmu_cfmuref_enable(void)
<b>Function descriptions</b>	enable CFMUREF pin input
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CFMUREF pin input */
cfmu_cfmuref_enable();
```

## cfmu\_cfmuref\_disable

The description of cfmu\_cfmuref\_disable is shown as below:

Table 3-141. Function `cfmu_cfmuref_disable`

Function name	<code>cfmu_cfmuref_disable</code>
Function prototype	<code>void cfmu_cfmuref_disable(void)</code>
Function descriptions	disable CFMUREF pin input
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CFMUREF pin input */
cfmu_cfmuref_disable();
```

### `cfmu_reference_signal_config`

The description of `cfmu_reference_signal_config` is shown as below:

Table 3-142. Function `cfmu_reference_signal_config`

Function name	<code>cfmu_reference_signal_config</code>
Function prototype	<code>void cfmu_reference_signal_config(uint32_t rssel)</code>
Function descriptions	configure the reference signal
Precondition	-
The called functions	-
Input parameter{in}	
<code>rssel</code>	reference signal select
<code>CFMU_RSSEL_CFMU_REF</code>	reference signal select the CFMUREF pin
<code>CFMU_RSSEL_INTERNAL_CLOCK</code>	reference signal select the internal clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the reference signal */
cfmu_reference_signal_config(CFMU_RSSEL_CFMUREF);
```

## cfmu\_digital\_filter\_config

The description of cfmu\_digital\_filter\_config is shown as below:

**Table 3-143. Function cfmu\_digital\_filter\_config**

<b>Function name</b>	cfmu_digital_filter_config
<b>Function prototype</b>	void cfmu_digital_filter_config(uint32_t dfssel)
<b>Function descriptions</b>	configure the digital filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dfssel</b>	digital filter select
<i>CFMU_DFSEL_DISABLE</i>	digital filtering is disabled
<i>CFMU_DFSEL_DIV1</i>	sampling clock for the digital filter is the frequency measuring clock divided by 1
<i>CFMU_DFSEL_DIV4</i>	sampling clock for the digital filter is the frequency measuring clock divided by 4
<i>CFMU_DFSEL_DIV16</i>	sampling clock for the digital filter is the frequency measuring clock divided by 16
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the digital filter */
```

```
cfmu_digital_filter_config(CFMU_DFSEL_DISABLE);
```

## cfmu\_reference\_clock\_config

The description of cfmu\_reference\_clock\_config is shown as below:

**Table 3-144. Function cfmu\_reference\_clock\_config**

<b>Function name</b>	cfmu_reference_clock_config
<b>Function prototype</b>	void cfmu_reference_clock_config(uint32_t rck_src, uint32_t rck_div, uint32_t val_edge)
<b>Function descriptions</b>	configure the measurement reference clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rck_src</b>	reference clock
<i>CFMU_RCKSRC_HXTAL</i>	HXTAL selected

<i>CFMU_RCKSRC_IRC3</i> <i>2M</i>	IRC32M selected
<i>CFMU_RCKSRC_IRC3</i> <i>2K</i>	IRC32K selected
<i>CFMU_RCKSRC_PCL</i> <i>K1</i>	PCLK1 selected
<b>Input parameter{in}</b>	
<b>rck_div</b>	rck divider
<i>CFMU_RCK_DIVx(x=3</i> <i>2,128,1024,8192)</i>	reference clock is divided by x
<b>Input parameter{in}</b>	
<b>val_edge</b>	valid edge
<i>CFMU_VAL_RISING</i>	rising edge valid
<i>CFMU_VAL_FALLING</i>	falling edge valid
<i>CFMU_VAL_BOTH</i>	rising and falling edge valid
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the measurement reference clock source */
```

```
cfmu_reference_clock_config(CFMU_RCKSRC_HXTAL,CFMU_RCK_DIV3,  
CFMU_VAL_RISING);
```

### cfmu\_measurement\_clock\_config

The description of cfmu\_measurement\_clock\_config is shown as below:

**Table 3-145. Function cfmu\_measurement\_clock\_config**

<b>Function name</b>	cfmu_measurement_clock_config
<b>Function prototype</b>	void cfmu_measurement_clock_config(uint32_t mck_src, uint32_t mck_div)
<b>Function descriptions</b>	configure the measurement target clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mck_src</b>	measurement clock
<i>CFMU_MCKSRC_HXTAL</i>	HXTAL selected
<i>CFMU_MCKSRC_IRC3</i> <i>2M</i>	IRC32M selected
<i>CFMU_MCKSRC_IRC3</i> <i>2K</i>	IRC32K selected

<i>CFMU_MCKSRC_PCLK1</i>	PCLK1 selected
<b>Input parameter{in}</b>	
<b>mck_div</b>	mck divider
<i>CFMU_MCK_DIVx(x=1, 4,8,32)</i>	MCK is divided by x
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the measurement target clock source */
```

```
cfmu_measurement_clock_config(CFMU_MCKSRC_HXTAL, CFMU_MCK_DIV1);
```

### cfmu\_limit\_value\_config

The description of cfmu\_limit\_value\_config is shown as below:

**Table 3-146. Function cfmu\_limit\_value\_config**

<b>Function name</b>	cfmu_limit_value_config
<b>Function prototype</b>	void cfmu_limit_value_config(uint32_t cfmu_hlv, uint32_t cfmu_llv)
<b>Function descriptions</b>	configure CFMU higher-limit value and lower-limit value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cfmu_hlv</b>	higher-limit value
<b>Input parameter{in}</b>	
<b>cfmu_llv</b>	lower-limit value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CFMU higher-limit value and lower-limit value */
```

```
cfmu_limit_value_config(0xFF,0x10);
```

### cfmu\_interrupt\_enable

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

**Table 3-147. Function cfmu\_interrupt\_enable**

<b>Function name</b>	cfmu_interrupt_enable
----------------------	-----------------------

<b>Function prototype</b>	void cfmu_interrupt_enable(cfmu_int_enum cfmu_interrupt)
<b>Function descriptions</b>	enable the CFMU interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cfmu_interrupt</b>	refer to <a href="#">Table 3-134. Enum cfmu_int_enum</a>
<i>CFMU_INT_OVF</i>	overflow interrupt request
<i>CFMU_INT_CFMEND</i>	measurement end interrupt request
<i>CFMU_INT_CFERR</i>	frequency error interrupt request
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the CFMU interrupt */
cfmu_interrupt_enable(CFMU_INT_OVF);
```

### cfmu\_interrupt\_disable

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

**Table 3-148. Function cfmu\_interrupt\_disable**

<b>Function name</b>	cfmu_interrupt_disable
<b>Function prototype</b>	void cfmu_interrupt_disable(cfmu_int_enum cfmu_interrupt)
<b>Function descriptions</b>	disable the CFMU interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cfmu_interrupt</b>	refer to <a href="#">Table 3-134. Enum cfmu_int_enum</a>
<i>CFMU_INT_OVF</i>	overflow interrupt request
<i>CFMU_INT_CFMEND</i>	measurement end interrupt request
<i>CFMU_INT_CFERR</i>	frequency error interrupt request
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the CFMU interrupt */
cfmu_interrupt_disable(CFMU_INT_OVF);
```

## cfmu\_interrupt\_flag\_get

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

**Table 3-149. Function cfmu\_interrupt\_flag\_get**

<b>Function name</b>	cfmu_interrupt_flag_get
<b>Function prototype</b>	FlagStatus cfmu_interrupt_flag_get(cfmu_int_flag_enum cfmu_int_flag)
<b>Function descriptions</b>	get the CFMU interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cfmu_int_flag</b>	interrupt flags, refer to <a href="#">Table 3-135. Enum cfmu_int_flag_enum</a>
CFMU_INT_FLAG_OVF	overflow flag
CFMU_INT_FLAG_CFMEND	clock frequency accuracy measurement end flag
CFMU_INT_FLAG_CFEERR	clock frequency error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the CFMU interrupt flags */
cfmu_interrupt_flag_get(CFMU_INT_FLAG_OVF);
```

## cfmu\_interrupt\_flag\_clear

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

**Table 3-150. Function cfmu\_interrupt\_flag\_clear**

<b>Function name</b>	cfmu_interrupt_flag_clear
<b>Function prototype</b>	void cfmu_interrupt_flag_clear(cfmu_int_flag_clear_enum cfmu_int_flag_clear)
<b>Function descriptions</b>	clear the CFMU interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cfmu_int_flag</b>	interrupt flags, refer to <a href="#">Table 3-136. Enum cfmu_int_flag_clear_enum</a>
CFMU_INT_FLAG_OVF_CLR	overflow flag clear
CFMU_INT_FLAG_CFMEND_CLR	clock frequency accuracy measurement end flag clear



CFMU_INT_FLAG_CFE RR_CLR	clock frequency error flag clear
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the CFMU interrupt flags */
```

```
cfmu_interrupt_flag_clear(CFMU_INT_FLAG_OVF_CLR);
```

## 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 3-151. CMP registers**

Registers	Descriptions
CMP_STAT	CMP status register
CMP_IFC	CMP interrupt flag clear register
CMP0_CS	CMP0 control and status register
CMP1_CS	CMP1 control and status register
CMP2_CS	CMP2 control and status register
CMP3_CS	CMP3 control and status register

### 3.5.2. Descriptions of Peripheral functions

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

**Table 3-152. 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_digital_filter_init	CMP digital filter init

Function name	Function description
cmp_enable	enable CMP
cmp_disable	disable CMP
cmp_lock_enable	lock the CMP
cmp_output_to_pin_enable	enable CMP output to pin
cmp_output_to_pin_disable	disable CMP output to pin
cmp_output_enable	enable CMP output
cmp_output_disable	disable CMP output
cmp_output_level_get	get output level
cmp_flag_get	get CMP flag
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 3-153. Enum cmp\_enum**

Member name	Function description
CMP0	comparator 0
CMP1	comparator 1
CMP2	comparator 2
CMP3	comparator 3

## cmp\_deinit

The description of cmp\_deinit is shown as below:

**Table 3-154. Function cmp\_deinit**

<b>Function name</b>	cmp_deinit
<b>Function prototype</b>	void cmp_deinit(cmp_enum cmp_periph);
<b>Function descriptions</b>	CMP deinit
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize CMP0 */
```

```
cmp_deinit(CMP0);
```

## cmp\_mode\_init

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

**Table 3-155. Function cmp\_mode\_init**

<b>Function name</b>	cmp_mode_init
<b>Function prototype</b>	void cmp_mode_init(cmp_enum cmp_periph, uint32_t inverting_input, uint32_t output_hysteresis);
<b>Function descriptions</b>	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 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>inverting_input</b>	inverting input
CMP_INVERTING_INP UT_NONE	No inverting input
CMP_INVERTING_INP UT_PC11	PC11 input
CMP_INVERTING_INP UT_PC12	PC12 input
CMP_INVERTING_INP UT_DAC0_OUT0	DAC0_OUT0 input
CMP_INVERTING_INP UT_DAC0_OUT1	DAC0_OUT1 input
<b>Input parameter{in}</b>	
<b>output_hysteresis</b>	hysteresis level
CMP_HYSTERESIS_N O	output no hysteresis
CMP_HYSTERESIS_L OW	output low hysteresis
CMP_HYSTERESIS_M IDDLE	output middle hysteresis
CMP_HYSTERESIS_HI GH	output high hysteresis
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
cmp_mode_init(CMP0, CMP_INVERTING_INPUT_PC11, CMP_HYSTERESIS_NO);
```

### cmp\_noninverting\_input\_select

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

**Table 3-156. 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 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>noninverting_input</b>	noninverting input select
CMP_NONINVERTING_INPUT_NONE	No noninverting input
CMP_NONINVERTING_INPUT_PC0_PC1_PC2_PC3	CMP noninverting input PC0 for CMP0 or PC1 for CMP1 or PC2 for CMP2 or PC3 for CMP3
CMP_NONINVERTING_INPUT_PC4_PC5_PC6_PD2	CMP noninverting input PC4 for CMP0 or PC5 for CMP1 or PC6 for CMP2 or PD3 for CMP3
CMP_NONINVERTING_INPUT_PC10_PC11_PC12_PD4	CMP noninverting input PC10 for CMP0 or PC11 for CMP1 or PC12 for CMP2 or PD4 for CMP3
CMP_NONINVERTING_INPUT_PC7_PC0_PC3_PD5	CMP noninverting input PC7 for CMP0 or PC0 for CMP1 or PC3 for CMP2 or PD5 for CMP3
CMP_NONINVERTING_INPUT_PC10_PC9	CMP noninverting input PC10 for CMP2 or PC9 for CMP3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* selecte the noninverting input for CMP0 */
cmp_noninverting_input_select(CMP0,
CMP_NONINVERTING_INPUT_PC0_PC1_PC2_PC);
```

## cmp\_output\_init

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

**Table 3-157. 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>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>output_polarity</b>	CMP output polarity
CMP_OUTPUT_POLARITY_INVERTED	output is inverted
CMP_OUTPUT_POLARITY_NONINVERTED	output is not inverted
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
cmp_output_init(CMP0, CMP_OUTPUT_POLARITY_NONINVERTED);
```

## cmp\_digital\_filter\_init

The description of cmp\_digital\_filter\_init is shown as below:

**Table 3-158. Function cmp\_digital\_filter\_init**

<b>Function name</b>	cmp_digital_filter_init
<b>Function prototype</b>	void cmp_digital_filter_init(cmp_enum cmp_periph, uint32_t sampling_frequency, uint32_t sampling_number);
<b>Function descriptions</b>	CMP digital filter init
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>sampling_frequen</b>	sampling frequency

<b>cy</b>	
<i>CMP_DIGITAL_FILTER_NOT_USED</i>	digital filter is not used
<i>CMP_SAMPLING_FREQUENCY_DIV8</i>	sampling frequency is $f_{CK\_CMP}/8$
<i>CMP_SAMPLING_FREQUENCY_DIV16</i>	sampling frequency is $f_{CK\_CMP}/16$
<i>CMP_SAMPLING_FREQUENCY_DIV32</i>	sampling frequency is $f_{CK\_CMP}/32$
<b>Input parameter{in}</b>	
<b>sampling_number</b>	sampling number
<i>CMP_SAMPLING_NUM_3_TIMES</i>	sampling number is 3 times
<i>CMP_SAMPLING_NUM_4_TIMES</i>	sampling number is 4 times
<b>输出参数{out}</b>	
-	-
<b>返回值</b>	
-	-

Example:

```
/* initialize CMP0 digital filter function */
```

```
cmp_digital_filter_init(CMP0,                                CMP_SAMPLING_FREQUENCY_DIV8,
CMP_SAMPLING_NUM_3_TIMES);
```

### cmp\_enable

The description of cmp\_enable is shown as below:

**Table 3-159. 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 3-153. 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 3-160. Function cmp\_disable**

Function name	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 3-153. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### cmp\_lock\_enable

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

**Table 3-161. Function cmp\_lock\_enable**

Function name	cmp_lock_enable
Function prototype	void cmp_lock_enable(cmp_enum cmp_periph);
Function descriptions	lock the comparator
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock CMP0 register */
cmp_lock_enable(CMP0);
```

## cmp\_output\_to\_pin\_enable

The description of cmp\_output\_to\_pin\_enable is shown as below:

**Table 3-162. Function cmp\_output\_to\_pin\_enable**

<b>Function name</b>	cmp_output_to_pin_enable
<b>Function prototype</b>	void cmp_output_to_pin_enable(cmp_enum cmp_periph);
<b>Function descriptions</b>	disable cmp output to pin
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CMP0 output to pin */
cmp_output_to_pin_enable(CMP0);
```

## cmp\_output\_to\_pin\_disable

The description of cmp\_output\_to\_pin\_disable is shown as below:

**Table 3-163. Function cmp\_output\_to\_pin\_disable**

<b>Function name</b>	cmp_output_to_pin_disable
<b>Function prototype</b>	void cmp_output_to_pin_disable(cmp_enum cmp_periph);
<b>Function descriptions</b>	disable cmp output to pin
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CMP0 output to pin */
```



cmp\_output\_to\_pin\_disable(CMP0);

## cmp\_output\_enable

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

**Table 3-164. Function cmp\_output\_enable**

Function name	cmp_output_enable
Function prototype	void cmp_output_enable(cmp_enum cmp_periph);
Function descriptions	disable cmp output
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CMP0 output to pin */
cmp_output_enable(CMP0);
```

## cmp\_output\_disable

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

**Table 3-165. Function cmp\_output\_disable**

Function name	cmp_output_disable
Function prototype	void cmp_output_disable(cmp_enum cmp_periph);
Function descriptions	disable cmp output
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CMP0 output */
```

```
cmp_output_disable(CMP0);
```

### cmp\_output\_level\_get

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

**Table 3-166. Function cmp\_output\_level\_get**

<b>Function name</b>	cmp_output_level_get
<b>Function prototype</b>	uint32_t cmp_output_level_get(uint32_t cmp_periph);
<b>Function descriptions</b>	get output level
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<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 3-167. 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>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	CMP interrupt flag
<b>CMP_FLAG_COMPAR</b>	CMP compare flag

<i>E</i>	
Output parameter{out}	
-	-
Return value	
FlagStatus	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 3-168. Function cmp\_flag\_clear**

Function name	cmp_flag_clear
Function prototype	void cmp_flag_clear(cmp_enum cmp_periph, uint32_t flag);
Function descriptions	clear CMP flag
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-153. Enum cmp_enum</a>
Input parameter{in}	
flag	CMP interrupt flag
CMP_FLAG_COMPAR <i>E</i>	CMP compare flag
Output parameter{out}	
-	-
Return value	
-	-

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 3-169. Function cmp\_interrupt\_enable**

Function name	cmp_interrupt_enable
Function prototype	void cmp_interrupt_enable(cmp_enum cmp_periph, uint32_t trigger_mode, 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 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>trigger_mode</b>	trigger mode
<i>CMP_INT_RISING_EDGE</i>	rising edge
<i>CMP_INT_FALLING_EDGE</i>	falling edge
<i>CMP_INT_BOTH_EDGES</i>	both edge
<b>Input parameter{in}</b>	
<b>interrupt</b>	CMP interrupt
<i>CMP_INT_COMPARE</i>	CMP compare interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the CMP0 interrupt */
```

```
cmp_interrupt_enable(CMP0, CMP_INT_RISING_EDGE, CMP_INT_COMPARE);
```

### cmp\_interrupt\_disable

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

**Table 3-170. 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 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>interrupt</b>	CMP interrupt
<i>CMP_INT_COMPARE</i>	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 3-171. 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 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	CMP interrupt flag
<b>CMP_INT_FLAG_COMPARE</b>	CMP compare interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-172. Function cmp\_interrupt\_flag\_clear**

<b>Function name</b>	cmp_interrupt_flag_clear
<b>Function prototype</b>	void cmp_interrupt_flag_clear(cmp_enum cmp_periph, uint32_t flag);
<b>Function descriptions</b>	clear 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 3-153. Enum cmp_enum</a>
<b>Input parameter{in}</b>	

<b>flag</b>	CMP interrupt flag
<i>CMP_INT_FLAG_COMPARE</i>	CMP compare interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

## 3.6. CPTIMER

The compare timer module (CPTIMER0/1) has two independent 16-bit unsigned counters, two modules total 4 counters. The compare timer can be configured to generate interrupt or EVIC request at set period. The CPTIMER registers are listed in chapter [3.6.1](#), the CPTIMER firmware functions are introduced in chapter [3.6.2](#).

### 3.6.1. Descriptions of Peripheral registers

CPTIMER registers are listed in the table shown as below:

**Table 3-173. CPTIMER registers**

Registers	Descriptions
CPTIMER_CTL0	CPTIMER control register
CPTIMER_CNT0	CPTIMER counter 0 register
CPTIMER_CNT1	CPTIMER counter 1 register
CPTIMER_CNT0PSC	CPTIMER counter 0 prescaler register
CPTIMER_CNT1PSC	CPTIMER counter 1 prescaler register
CPTIMER_INTEN	CPTIMER interrupt enable register
CPTIMER_INTF	CPTIMER interrupt flag register
CPTIMER_SWEVG	CPTIMER software event generation register
CPTIMER_CNT0CAR	CPTIMER counter 0 autoreload register
CPTIMER_CNT1CAR	CPTIMER counter 1 autoreload register

### 3.6.2. Descriptions of Peripheral functions

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

**Table 3-174. CPTIMER firmware function**

Function name	Function description
cptimer_deinit	CPTIMER deinit

Function name	Function description
cptimer_enable	enable CPTIMER
cptimer_disable	disable CPTIMER
cptimer_prescaler_config	configure CPTIMER prescaler
cptimer_autoreload_value_config	configure CPTIMER autoreload value
cptimer_counter_value_config	configure CPTIMER counter value
cptimer_counter_read	read CPTIMER counter value
cptimer_prescaler_read	read CPTIMER prescaler value
cptimer_event_software_generate	software generate events
cptimer_flag_get	get CPTIMER flag
cptimer_flag_clear	clear CPTIMER flag
cptimer_interrupt_enable	enable CPTIMER interrupt
cptimer_interrupt_disable	disable CPTIMER interrupt
cptimer_interrupt_flag_get	get CPTIMER interrupt flag
cptimer_interrupt_flag_clear	clear CPTIMER interrupt flag

### cptimer\_deinit

The description of cptimer\_deinit is shown as below:

**Table 3-175. Function cptimer\_deinit**

Function name	cptimer_deinit
Function prototype	void cptimer_deinit(uint32_t cptimer_periph);
Function descriptions	CPTIMER deinit
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
cptimer_periph	cptimer peripheral
CPTIMERx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize CPTIMER0 */
cptimer_deinit(CPTIMER0);
```

### cptimer\_enable

The description of cptimer\_enable is shown as below:

**Table 3-176. Function cptimer\_enable**

Function name	cptimer_enable
Function prototype	void cptimer_enable(uint32_t cptimer_periph);

<b>Function descriptions</b>	enable CPTIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CPTIMER0 */
cptimer_enable(CPTIMER0);
```

### **cptimer\_disable**

The description of cptimer\_disable is shown as below:

**Table 3-177. Function cptimer\_disable**

<b>Function name</b>	cptimer_disable
<b>Function prototype</b>	void cptimer_disable(uint32_t cptimer_periph);
<b>Function descriptions</b>	disable CPTIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CPTIMER0 */
cptimer_disable(CPTIMER0);
```

### **cptimer\_prescaler\_config**

The description of cptimer\_prescaler\_config is shown as below:

**Table 3-178. Function cptimer\_prescaler\_config**

<b>Function name</b>	cptimer_prescaler_config
<b>Function prototype</b>	void cptimer_prescaler_config(uint32_t cptimer_periph , uint32_t counter, uint16_t prescaler);



<b>Function descriptions</b>	configure the CPTIMER prescaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>counter</b>	cptimer counter
CPTIMER_COUNTERx	x=0,1
<b>Input parameter{in}</b>	
<b>prescaler</b>	prescaler value,0~65535
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CPTIMER0 counter 0 prescaler */
cptimer_prescaler_config(CPTIMER0, CPTIMER_COUNTER0, 10U);
```

### **cptimer\_autoreload\_value\_config**

The description of cptimer\_autoreload\_value\_config is shown as below:

**Table 3-179. Function cptimer\_autoreload\_value\_config**

<b>Function name</b>	cptimer_autoreload_value_config
<b>Function prototype</b>	void cptimer_autoreload_value_config(uint32_t cptimer_periph , uint32_t counter, uint16_t autoreload);
<b>Function descriptions</b>	configure CPTIMER autoreload value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>counter</b>	cptimer counter
CPTIMER_COUNTERx	x=0,1
<b>Input parameter{in}</b>	
<b>autoreload</b>	autoreload value,0~65535
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* configure the CPTIMER0 counter 0 autoreload value */
cptimer_autoreload_value_config (CPTIMER0, CPTIMER_COUNTER0, 10U);
```

### **cptimer\_counter\_value\_config**

The description of cptimer\_counter\_value\_config is shown as below:

**Table 3-180. Function cptimer\_counter\_value\_config**

Function name	cptimer_counter_value_config
Function prototype	void cptimer_counter_value_config(uint32_t cptimer_periph, uint32_t counter, uint16_t value);
Function descriptions	configure CPTIMER counter register value
Precondition	-
The called functions	-
Input parameter{in}	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
Input parameter{in}	
<b>counter</b>	cptimer counter
CPTIMER_COUNT ERx	x=0,1
Input parameter{in}	
<b>value</b>	the counter value, 0~65535
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CPTIMER counter register value */
cptimer_counter_value_config (CPTIMER0, CPTIMER_COUNTER0, 10U);
```

### **cptimer\_counter\_read**

The description of cptimer\_counter\_read is shown as below:

**Table 3-181. Function cptimer\_counter\_read**

Function name	cptimer_counter_read
Function prototype	uint16_t cptimer_counter_read(uint32_t cptimer_periph, uint32_t counter);

<b>Function descriptions</b>	read CPTIMER counter value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>counter</b>	cptimer counter
CPTIMER_COUNT ERx	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	counter value (0~65535)

Example:

```
/* read CPTIMER counter value */
uint16_t cnt = cptimer_counter_read(CPTIMER0, CPTIMER_COUNTER0);
```

### **cptimer\_prescaler\_read**

The description of cptimer\_prescaler\_read is shown as below:

**Table 3-182. Function cptimer\_prescaler\_read**

<b>Function name</b>	cptimer_prescaler_read
<b>Function prototype</b>	uint16_t cptimer_prescaler_read(uint32_t cptimer_periph, uint32_t counter);
<b>Function descriptions</b>	read CPTIMER prescaler value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>counter</b>	cptimer counter
CPTIMER_COUNT ERx	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	prescaler value (0~65535)

Example:

```
/* read CPTIMER prescaler value */
uint16_t pre = cptimer_prescaler_read(CPTIMER0, CPTIMER_COUNTER0);
```

### **cptimer\_event\_software\_generate**

The description of cptimer\_event\_software\_generate is shown as below:

**Table 3-183. Function cptimer\_event\_software\_generate**

<b>Function name</b>	cptimer_event_software_generate
<b>Function prototype</b>	void cptimer_event_software_generate(uint32_t cptimer_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>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>event</b>	the timer software event generation sources
CPTIMER_EVENT_SR C_UPG0	counter 0 update event generation
CPTIMER_EVENT_SR C_UPG1	counter 1 update event generation
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* generate CPTIMER0 counter 0 update event */
cptimer_event_software_generate (CPTIMER0, CPTIMER_EVENT_SRC_UPG0);
```

### **cptimer\_flag\_get**

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

**Table 3-184. Function cptimer\_flag\_get**

<b>Function name</b>	cptimer_flag_get
<b>Function prototype</b>	FlagStatus cptimer_flag_get(uint32_t cptimer_periph, uint32_t flag);
<b>Function descriptions</b>	get CPTIMER flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral

CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>flag</b>	the CPTIMER flag
CPTIMER_FLAG_CNT0UP	CPTIMER counter 0 update flag
CPTIMER_FLAG_CNT1UP	CPTIMER counter 1 update flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the CPTIMER0 flag bit */
FlagStatus flag_value;
flag_value = cptimer_flag_get(CPTIMER0, CPTIMER_FLAG_CNT1UP);
```

### **cptimer\_flag\_clear**

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

**Table 3-185. Function cptimer\_flag\_clear**

<b>Function name</b>	cptimer_flag_clear
<b>Function prototype</b>	void cptimer_flag_clear(uint32_t cptimer_periph, uint32_t flag);
<b>Function descriptions</b>	clear CPTIMER flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>flag</b>	the CPTIMER flag
CPTIMER_FLAG_CNT0UP	CPTIMER counter 0 update flag
CPTIMER_FLAG_CNT1UP	CPTIMER counter 1 update flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the CPTIMER0 flag bit */
cptimer_flag_clear(CPTIMER0, CPTIMER_FLAG_CNT0UP);
```

## cptimer\_interrupt\_enable

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

**Table 3-186. Function cptimer\_interrupt\_enable**

<b>Function name</b>	cptimer_interrupt_enable
<b>Function prototype</b>	void cptimer_interrupt_enable(uint32_t cptimer_periph, uint32_t interrupt);
<b>Function descriptions</b>	enable CPTIMER interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>interrupt</b>	CPTIMER interrupt
CPTIMER_INT_CNT0UP	CPTIMER counter 0 update interrupt
CPTIMER_INT_CNT1UP	CPTIMER counter 1 update interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the CPTIMER0 interrupt */
cptimer_interrupt_enable(CPTIMER0, CPTIMER_INT_CNT1UP);
```

## cptimer\_interrupt\_disable

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

**Table 3-187. Function cptimer\_interrupt\_disable**

<b>Function name</b>	cptimer_interrupt_disable
<b>Function prototype</b>	void cptimer_interrupt_disable(uint32_t cptimer_periph, uint32_t interrupt);
<b>Function descriptions</b>	disable CPTIMER interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
<b>Input parameter{in}</b>	
<b>interrupt</b>	CPTIMER interrupt
CPTIMER_INT_CNT0UP	CPTIMER counter 0 update interrupt
P	

CPTIMER_INT_CNT1UP	CPTIMER counter 1 update interrupt
P	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the CPTIMER0 interrupt */
cptimer_interrupt_disable(CPTIMER0, CPTIMER_INT_CNT1UP);
```

### **cptimer\_interrupt\_flag\_get**

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

**Table 3-188. Function cptimer\_interrupt\_flag\_get**

<b>Function name</b>	cptimer_interrupt_flag_get
<b>Function prototype</b>	FlagStatus cptimer_interrupt_flag_get(uint32_t cptimer_periph, uint32_t int_flag);
<b>Function descriptions</b>	get CPTIMER interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>cptimer_periph</b>	cptimer peripheral
CPTIMERx	x=0,1
Input parameter{in}	
<b>int_flag</b>	CPTIMER interrupt flag
CPTIMER_INT_FLAG_CNT0UP	CPTIMER counter 0 update interrupt flag
CPTIMER_INT_FLAG_CNT1UP	CPTIMER counter 1 update interrupt flag
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the CPTIMER0 interrupt bit*/
FlagStatus flag_value;
flag_value = cptimer_interrupt_flag_get(CPTIMER0, CPTIMER_INT_FLAG_CNT0UP);
```

### **cptimer\_interrupt\_flag\_clear**

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

Table 3-189. Function `cptimer_interrupt_flag_clear`

<b>Function name</b>	<code>cptimer_interrupt_flag_clear</code>
<b>Function prototype</b>	<code>void cptimer_interrupt_flag_clear(uint32_t cptimer_periph, uint32_t int_flag);</code>
<b>Function descriptions</b>	clear CPTIMER interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><code>cptimer_periph</code></b>	cptimer peripheral
<code>CPTIMERx</code>	<code>x=0,1</code>
<b>Input parameter{in}</b>	
<b><code>int_flag</code></b>	CPTIMER interrupt flag
<code>CPTIMER_INT_FLAG_CNT0UP</code>	CPTIMER counter 0 update interrupt flag
<code>CPTIMER_INT_FLAG_CNT1UP</code>	CPTIMER counter 1 update interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the CPTIMER0 interrupt bit*/
cptimer_interrupt_flag_clear(CPTIMER0, CPTIMER_INT_FLAG_CNT0UP);
```

## 3.7. CPTIMERW

The compare timer W module contains a 32-bit unsigned counter which can be configured to 16-bit. The compare timer W has two input capture channels and four compare match channels with 2 channels pulse output supported. All the channels and counter update event can be configured to generate interrupt or EVIC event at set period. The CPTIMERW registers are listed in chapter [3.7.1](#), the CPTIMERW firmware functions are introduced in chapter [3.7.2](#).

### 3.7.1. Descriptions of Peripheral registers

CPTIMER registers are listed in the table shown as below:

Table 3-190. CPTIMERW registers

Registers	Descriptions
<code>CPTIMERW_CTL0</code>	CPTIMERW control 0 register
<code>CPTIMERW_CTL1</code>	CPTIMERW control 1 register
<code>CPTIMERW_PSC</code>	CPTIMERW prescaler register
<code>CPTIMERW_CNT</code>	CPTIMERW counter register
<code>CPTIMERW_INTEN</code>	CPTIMERW interrupt enable register



Registers	Descriptions
CPTIMERW_INTF	CPTIMERW interrupt flag register
CPTIMERW_SWEVG	CPTIMERW software event generation register
CPTIMERW_ICH0CV	CPTIMERW input channel 0 capture value register
CPTIMERW_ICH1CV	CPTIMERW input channel 1 capture value register
CPTIMERW_OCH0CV	CPTIMERW output compare channel 0 compare value register
CPTIMERW_OCH1CV	CPTIMERW output compare channel 1 compare value register
CPTIMERW_OCH2CV	CPTIMERW output compare channel 2 compare value register
CPTIMERW_OCH3CV	CPTIMERW output compare channel 3 compare value register
CPTIMERW_CAR	CPTIMERW autoreload register

### 3.7.2. Descriptions of Peripheral functions

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

**Table 3-191. CPTIMER firmware function**

Function name	Function description
cptimerw_deinit	CPTIMER deinit
cptimerw_struct_para_init	initialize CPTIMERW init parameter struct
cptimerw_enable	enable CPTIMER
cptimerw_disable	disable CPTIMER
cptimerw_prescaler_config	configure CPTIMER prescaler
cptimerw_autoreload_value_config	configure CPTIMER autoreload value
cptimerw_counter_value_config	configure CPTIMER counter value
cptimerw_counter_clear_source_config	configure CPTIMER counter clear source
cptimerw_counter_read	read CPTIMER counter value
cptimerw_prescaler_read	read CPTIMER prescaler value
cptimerw_channel_output_mode_select	select CPTIMERW channel output compare mode
cptimerw_channel_output_compare_value_config	configure CPTIMERW channel output compare value
cptimerw_channel_output_state_config	configure CPTIMERW output compare channel enable state
cptimerw_channel_input_struct_para_init	initialize CPTIMERW channel input capture parameter struct
cptimerw_input_capture_config	configure CPTIMERW input capture parameter
cptimerw_channel_capture_value_register_read	read CPTIMERW channel input capture value
cptimerw_event_software_generate	software generate events
cptimerw_flag_get	get CPTIMER flag
cptimerw_flag_clear	clear CPTIMER flag
cptimerw_interrupt_enable	enable CPTIMER interrupt
cptimerw_interrupt_disable	disable CPTIMER interrupt
cptimerw_interrupt_flag_get	get CPTIMER interrupt flag
cptimerw_interrupt_flag_clear	clear CPTIMER interrupt flag

### Structure `cptimerw_init_parameter_struct`

Table 3-192. `cptimerw_init_parameter_struct`

Member name	Function description
width	counter bit width (CPTIMERW_CNT_WIDTH_16BIT, CPTIMERW_CNT_WIDTH_32BIT)
prescaler	counter prescaler value (0~0xFFFF)
period	counter autoreload value (16bit:0~0xFFFF, 32-bit:0~0xFFFFFFFF)
clear_source	counter clear source (CPTIMERW_CNT_CLEAR_DISABLE, CPTIMERW_CNT_CLEAR_ICH0, CPTIMERW_CNT_CLEAR_ICH1)
clockdivision	input capture clock division value (CPTIMERW_CKDIV_DIV1, CPTIMERW_CKDIV_DIV2, CPTIMERW_CKDIV_DIV4)

### Structure `cptimerw_ic_parameter_struct`

Table 3-193. `cptimerw_ic_parameter_struct`

Member name	Function description
icedge	channel input edge (CPTIMERW_IC_FALLING_EDGE, CPTIMERW_IC_RISING_EDGE, CPTIMERW_IC_BOTH_EDGE, CPTIMERW_IC_DISABLE)
icfilter	channel input capture filter control (0~15)

### `cptimerw_deinit`

The description of `cptimerw_deinit` is shown as below:

Table 3-194. Function `cptimerw_deinit`

Function name	<code>cptimerw_deinit</code>
Function prototype	<code>void cptimerw_deinit(void);</code>
Function descriptions	CPTIMERW deinit
Precondition	-
The called functions	<code>rcu_periph_reset_enable</code> / <code>rcu_periph_reset_disable</code>
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* deinitialize CPTIMERW0 */
cptimerw_deinit(CPTIMERW0);
```

### **cptimerw\_struct\_para\_init**

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

**Table 3-195. Function cptimerw\_struct\_para\_init**

<b>Function name</b>	cptimerw_struct_para_init
<b>Function prototype</b>	void cptimerw_struct_para_init(cptimerw_init_parameter_struct *initpara)
<b>Function descriptions</b>	initialize CPTIMER init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>initpara</b>	refer to <a href="#">Table 3-192. cptimerw_init_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* init parameter struct with a default value */
cptimerw_parameter_struct para;
cptimerw_struct_para_init(&para);
```

### **cptimerw\_init**

The description of cptimerw\_init is shown as below:

**Table 3-196. Function cptimerw\_init**

<b>Function name</b>	cptimerw_init
<b>Function prototype</b>	void cptimerw_init(cptimerw_init_parameter_struct *initpara)
<b>Function descriptions</b>	initialize CPTIMERW counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>initpara</b>	refer to <a href="#">Table 3-192. cptimerw_init_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize CPTIMERW counter */
cptimerw_parameter_struct para;
para.width = CPTIMERW_CNT_WIDTH_16BIT;
```

```
para.period = 10000U;
para.prescaler = 99;
para.clear_source = CPTIMERW_CNT_CLEAR_DISABLE;
para.clockdivision = CPTIMERW_CKDIV_DIV1;
cptimerw_init(&para);
```

### **cptimerw\_autoreload\_value\_config**

The description of cptimerw\_autoreload\_value\_config is shown as below:

**Table 3-197. Function cptimerw\_autoreload\_value\_config**

<b>Function name</b>	cptimerw_autoreload_value_config
<b>Function prototype</b>	void cptimerw_autoreload_value_config(uint32_t autoreload);
<b>Function descriptions</b>	configure CPTIMERW autoreload value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>autoreload</b>	the counter autoreload value, 16 or 32-bit based on counter width
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CPTIMERW autoreload value */
cptimerw_autoreload_value_config (10000U);
```

### **cptimerw\_prescaler\_config**

The description of cptimerw\_prescaler\_config is shown as below:

**Table 3-198. Function cptimerw\_lock\_enable**

<b>Function name</b>	cptimerw_prescaler_config
<b>Function prototype</b>	void cptimerw_prescaler_config(uint16_t prescaler);
<b>Function descriptions</b>	configure the CPTIMERW prescaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>prescaler</b>	prescaler value, 0~65535
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CPTIMERW prescaler */
cptimerw_prescaler_config(99U);
```

### **cptimerw\_counter\_value\_config**

The description of cptimerw\_counter\_value\_config is shown as below:

**Table 3-199. Function cptimerw\_counter\_value\_config**

<b>Function name</b>	cptimerw_counter_value_config
<b>Function prototype</b>	void cptimerw_counter_value_config(uint16_t value);
<b>Function descriptions</b>	configure CPTIMERW counter register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>value</b>	the counter value, 16 or 32-bit based on counter width
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CPTIMERW counter register value */
cptimerw_counter_value_config (10U);
```

### **cptimerw\_counter\_clear\_source\_config**

The description of cptimerw\_counter\_clear\_source\_config is shown as below:

**Table 3-200. Function cptimerw\_counter\_clear\_source\_config**

<b>Function name</b>	cptimerw_counter_clear_source_config
<b>Function prototype</b>	void cptimerw_counter_clear_source_config(uint32_t clear_source);
<b>Function descriptions</b>	configure CPTIMERW counter clear source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clear_source</b>	the counter clear source
CPTIMERW_CNT_CLEAR_DISABLE	counter clear disable
CPTIMERW_CNT_CLEAR_ICH0	counter clear at channel 0 input capture event

CPTIMERW_CNT_CLEAR_ICH1	counter clear at channel 1 input capture event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* onfigure CPTIMERW counter clear source */
cptimerw_counter_clear_source_config(CPTIMERW_CNT_CLEAR_ICH0);
```

### **cptimerw\_counter\_read**

The description of cptimerw\_counter\_read is shown as below:

**Table 3-201. Function cptimerw\_counter\_read**

Function name	cptimerw_counter_read
Function prototype	uint32_t cptimerw_counter_read(void);
Function descriptions	read CPTIMERW counter value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	counter value

Example:

```
/* read CPTIMERW counter value */
uint32_t cnt = cptimerw_counter_read();
```

### **cptimerw\_prescaler\_read**

The description of cptimerw\_prescaler\_read is shown as below:

**Table 3-202. Function cptimerw\_output\_disable**

Function name	cptimerw_prescaler_read
Function prototype	uint16_t cptimerw_prescaler_read(void);
Function descriptions	read CPTIMERW prescaler value
Precondition	-
The called	-

<b>functions</b>	
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint16_t	prescaler value (0~65535)

Example:

```
/* read CPTIMERW prescaler value */
uint16_t pre = cptimerw_prescaler_read(void);
```

### **cptimerw\_enable**

The description of cptimerw\_enable is shown as below:

**Table 3-203. Function cptimerw\_enable**

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

Example:

```
/* enable CPTIMERW */
cptimerw_enable(CPTIMERW);
```

### **cptimerw\_disable**

The description of cptimerw\_disable is shown as below:

**Table 3-204. Function cptimerw\_disable**

<b>Function name</b>	cptimerw_disable
<b>Function prototype</b>	void cptimerw_disable(void);
<b>Function descriptions</b>	disable CPTIMERW
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable CPTIMERW */
cptimerw_disable(CPTIMERW);
```

### **cptimerw\_channel\_output\_mode\_select**

The description of cptimerw\_channel\_output\_mode\_select is shown as below:

**Table 3-205. Function cptimerw\_channel\_output\_mode\_select**

Function name	cptimerw_channel_output_mode_select
Function prototype	void cptimerw_channel_output_mode_select(uint32_t channel, uint32_t mode);
Function descriptions	select CPTIMERW channel output compare mode
Precondition	-
The called functions	-
Input parameter{in}	
channel	output compare channel
CPTIMERW_OCHx	x=0,1
Input parameter{in}	
mode	output compare mode
CPTIMERW_OC_MOD E_DISABLE	output disable
CPTIMERW_OC_MOD E_MANTAIN	output mantain
CPTIMERW_OC_MOD E_LOW_TOGGLE	initial level is low and toggle on compare match
CPTIMERW_OC_MOD E_HIGH_TOGGLE	initial level is high and toggle on compare match
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select CPTIMERW channel output compare mode */
cptimerw_channel_output_mode_select(CPTIMERW_OCH0,
CPTIMERW_OC_MODE_LOW_TOGGLE);
```



## cptimerw\_channel\_output\_compare\_value\_config

The description of cptimerw\_channel\_output\_compare\_value\_config is shown as below:

**Table 3-206. Function cptimerw\_channel\_output\_compare\_value\_config**

<b>Function name</b>	cptimerw_channel_output_compare_value_config
<b>Function prototype</b>	void cptimerw_channel_output_compare_value_config(uint32_t channel, uint32_t ochcv);
<b>Function descriptions</b>	configure CPTIMERW channel output compare value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channel</b>	output compare channel
CPTIMERW_OCHx	x= 0,1,2,3
<b>Input parameter{in}</b>	
<b>ochcv</b>	channel output compare value, 16 or 32-bit based on counter width
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CPTIMERW channel output compare value */
cptimerw_channel_output_compare_value_config(CPTIMERW_OCH0, 99U);
```

## cptimerw\_channel\_output\_state\_config

The description of cptimerw\_channel\_output\_state\_config is shown as below:

**Table 3-207. Function cptimerw\_channel\_output\_state\_config**

<b>Function name</b>	cptimerw_channel_output_state_config
<b>Function prototype</b>	void cptimerw_channel_output_state_config(uint32_t channel, uint32_t state);
<b>Function descriptions</b>	configure CPTIMERW output compare channel enable state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channel</b>	output compare channel
CPTIMERW_OCHx	x= 0,1,2,3
<b>Input parameter{in}</b>	
<b>state</b>	enable state
CPTIMERW_OCX_DISABLE	output compare channel disable
CPTIMERW_OCX_ENABLE	output compare channel enable

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CPTIMERW output compare channel enable state */
cptimerw_channel_output_state_config(CPTIMERW_OCH0, CPTIMERW_OCX_ENABLE);
```

### **cptimerw\_channel\_input\_struct\_para\_init**

The description of cptimerw\_channel\_input\_struct\_para\_init is shown as below:

**Table 3-208. Function cptimerw\_channel\_input\_struct\_para\_init**

Function name	cptimerw_channel_input_struct_para_init
Function prototype	void cptimerw_channel_input_struct_para_init(cptimerw_ic_parameter_struct *icpara)
Function descriptions	initialize CPTIMERW channel input capture parameter struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
icpara	refer to <a href="#">Table 3-193. cptimerw ic parameter struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* init parameter struct with a default value */
cptimerw_ic_parameter_struct para;
cptimerw_channel_input_struct_para_init(&para);
```

### **cptimerw\_input\_capture\_config**

The description of cptimerw\_input\_capture\_config is shown as below:

**Table 3-209. Function cptimerw\_input\_capture\_config**

Function name	cptimerw_input_capture_config
Function prototype	void cptimerw_input_capture_config(uint16_t channel, cptimerw_ic_parameter_struct *icpara)
Function descriptions	configure CPTIMERW input capture parameter
Precondition	-
The called functions	-

Input parameter{in}	
<b>channel</b>	Input capture channel
CPTIMERW_ICHx	x= 0,1
Input parameter{in}	
<b>initpara</b>	refer to <a href="#">Table 3-454. Structure gptimer_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CPTIMERW input capture parameter */
cptimerw_ic_parameter_struct para;
para.icedge = CPTIMERW_IC_RISING_EDGE;
para.icfilter = 5U;
cptimerw_input_capture_config(CPTIMERW_ICH0, &para);
```

### **cptimerw\_channel\_capture\_value\_register\_read**

The description of cptimerw\_channel\_capture\_value\_register\_read is shown as below:

**Table 3-210. Function cptimerw\_channel\_capture\_value\_register\_read**

<b>Function name</b>	cptimerw_channel_capture_value_register_read
<b>Function prototype</b>	uint32_t cptimerw_channel_capture_value_register_read(uint16_t channel);
<b>Function descriptions</b>	read CPTIMERW channel input capture value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>channel</b>	input capture channel
CPTIMERW_ICHx	x= 0,1
Output parameter{out}	
-	-
Return value	
<b>Uint32_t</b>	capture value

Example:

```
/* read CPTIMERW channel input capture value */
uint32_t val = cptimerw_channel_capture_value_register_read(CPTIMERW_ICH0);
```

### **cptimerw\_event\_software\_generate**

The description of cptimerw\_event\_software\_generate is shown as below:

Table 3-211. Function `cptimerw_event_software_generate`

<b>Function name</b>	<code>cptimerw_event_software_generate</code>
<b>Function prototype</b>	<code>void cptimerw_event_software_generate(uint32_t event);</code>
<b>Function descriptions</b>	software generate events
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>event</b>	the timer software event generation sources
<code>CPTIMERW_EVENT_S</code> <code>RC_UPG</code>	counter update event generation
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* generate counter update event */
cptimerw_event_software_generate (CPTIMERW_EVENT_SRC_UPG);
```

### `cptimerw_flag_get`

The description of `cptimerw_flag_get` is shown as below:

Table 3-212. Function `cptimerw_flag_get`

<b>Function name</b>	<code>cptimerw_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus cptimerw_flag_get(uint32_t flag);</code>
<b>Function descriptions</b>	get CPTIMERW flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	CPTIMERW interrupt flag
<code>CPTIMERW_FLAG_UP</code>	CPTIMERW update flag
<code>CPTIMERW_FLAG_IC</code> <code>H0</code>	CPTIMERW channel 0 input capture flag
<code>CPTIMERW_FLAG_IC</code> <code>H1</code>	CPTIMERW channel 1 input capture flag
<code>CPTIMERW_FLAG_OC</code> <code>H0</code>	CPTIMERW channel 0 output compare flag
<code>CPTIMERW_FLAG_OC</code> <code>H1</code>	CPTIMERW channel 1 output compare flag
<code>CPTIMERW_FLAG_OC</code> <code>H2</code>	CPTIMERW channel 2 output compare flag
<code>CPTIMERW_FLAG_OC</code> <code>H3</code>	CPTIMERW channel 3 output compare flag

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

Example:

```
/* get the CPTIMERW flag */
FlagStatus flag_value;
flag_value = cptimerw_flag_get(CPTIMERW_FLAG_OCH0);
```

### **cptimerw\_flag\_clear**

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

**Table 3-213. Function cptimerw\_flag\_clear**

Function name	cptimerw_flag_clear
Function prototype	void cptimerw_flag_clear(uint32_t flag);
Function descriptions	clear CPTIMERW flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	CPTIMERW interrupt flag
CPTIMERW_FLAG_UP	CPTIMERW update flag
CPTIMERW_FLAG_IC H0	CPTIMERW channel 0 input capture flag
CPTIMERW_FLAG_IC H1	CPTIMERW channel 1 input capture flag
CPTIMERW_FLAG_OC H0	CPTIMERW channel 0 output compare flag
CPTIMERW_FLAG_OC H1	CPTIMERW channel 1 output compare flag
CPTIMERW_FLAG_OC H2	CPTIMERW channel 2 output compare flag
CPTIMERW_FLAG_OC H3	CPTIMERW channel 3 output compare flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the CPTIMERW flag */
cptimerw_flag_clear(CPTIMERW_FLAG_UP);
```

## cptimerw\_interrupt\_enable

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

**Table 3-214. Function cptimerw\_interrupt\_enable**

<b>Function name</b>	cptimerw_interrupt_enable
<b>Function prototype</b>	void cptimerw_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable CPTIMERW interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	CPTIMERW interrupt
<i>CPTIMERW_INT_UP</i>	CPTIMERW update interrupt
<i>CPTIMERW_INT_ICH0</i>	CPTIMERW channel 0 input capture interrupt
<i>CPTIMERW_INT_ICH1</i>	CPTIMERW channel 1 input capture interrupt
<i>CPTIMERW_INT_OCH0</i>	CPTIMERW channel 0 output compare interrupt
<i>CPTIMERW_INT_OCH1</i>	CPTIMERW channel 1 output compare interrupt
<i>CPTIMERW_INT_OCH2</i>	CPTIMERW channel 2 output compare interrupt
<i>CPTIMERW_INT_OCH3</i>	CPTIMERW channel 3 output compare interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the CPTIMERW interrupt */
cptimerw_interrupt_enable(CPTIMERW_INT_UP);
```

## cptimerw\_interrupt\_disable

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

**Table 3-215. Function cptimerw\_interrupt\_disable**

<b>Function name</b>	cptimerw_interrupt_disable
<b>Function prototype</b>	void cptimerw_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable CPTIMERW interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

interrupt	CPTIMERW interrupt
<i>CPTIMERW_INT_UP</i>	CPTIMERW update interrupt
<i>CPTIMERW_INT_ICH0</i>	CPTIMERW channel 0 input capture interrupt
<i>CPTIMERW_INT_ICH1</i>	CPTIMERW channel 1 input capture interrupt
<i>CPTIMERW_INT_OCH0</i>	CPTIMERW channel 0 output compare interrupt
<i>CPTIMERW_INT_OCH1</i>	CPTIMERW channel 1 output compare interrupt
<i>CPTIMERW_INT_OCH2</i>	CPTIMERW channel 2 output compare interrupt
<i>CPTIMERW_INT_OCH3</i>	CPTIMERW channel 3 output compare interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the CPTIMERW interrupt */
cptimerw_interrupt_disable(CPTIMERW_INT_UP);
```

### **cptimerw\_interrupt\_flag\_get**

The description of `cptimerw_interrupt_flag_get` is shown as below:

**Table 3-216. Function `cptimerw_interrupt_flag_get`**

Function name	<code>cptimerw_interrupt_flag_get</code>
Function prototype	<code>FlagStatus cptimerw_interrupt_flag_get(uint32_t flag);</code>
Function descriptions	get CPTIMERW interrupt flag
Precondition	-
The called functions	-
<b>Input parameter{in}</b>	
int_flag	CPTIMERW interrupt flag
<i>CPTIMERW_INT_FLAG_UP</i>	CPTIMERW update interrupt flag
<i>CPTIMERW_INT_FLAG_ICH0</i>	CPTIMERW channel 0 input capture interrupt flag
<i>CPTIMERW_INT_FLAG_ICH1</i>	CPTIMERW channel 1 input capture interrupt flag
<i>CPTIMERW_INT_FLAG_OCH0</i>	CPTIMERW channel 0 output compare interrupt flag
<i>CPTIMERW_INT_FLAG_OCH1</i>	CPTIMERW channel 1 output compare interrupt flag

<i>G_OCH1</i>	
<i>CPTIMERW_INT_FLG</i> <i>G_OCH2</i>	CPTIMERW channel 2 output compare interrupt flag
<i>CPTIMERW_INT_FLG</i> <i>G_OCH3</i>	CPTIMERW channel 3 output compare interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```

/* get the CPTIMERW interrupt bit */
FlagStatus flag_value;
flag_value = cptimerw_interrupt_flag_get(CPTIMERW_INT_FLAG_UP);

```

### **cptimerw\_interrupt\_flag\_clear**

The description of `cptimerw_interrupt_flag_clear` is shown as below:

**Table 3-217. Function `cptimerw_interrupt_flag_clear`**

<b>Function name</b>	<code>cptimerw_interrupt_flag_clear</code>
<b>Function prototype</b>	<code>void cptimerw_interrupt_flag_clear(uint32_t flag);</code>
<b>Function descriptions</b>	clear CPTIMERW interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	CPTIMERW interrupt flag
<i>CPTIMERW_INT_FLG</i> <i>G_UP</i>	CPTIMERW update interrupt flag
<i>CPTIMERW_INT_FLG</i> <i>G_ICH0</i>	CPTIMERW channel 0 input capture interrupt flag
<i>CPTIMERW_INT_FLG</i> <i>G_ICH1</i>	CPTIMERW channel 1 input capture interrupt flag
<i>CPTIMERW_INT_FLG</i> <i>G_OCH0</i>	CPTIMERW channel 0 output compare interrupt flag
<i>CPTIMERW_INT_FLG</i> <i>G_OCH1</i>	CPTIMERW channel 1 output compare interrupt flag
<i>CPTIMERW_INT_FLG</i> <i>G_OCH2</i>	CPTIMERW channel 2 output compare interrupt flag
<i>CPTIMERW_INT_FLG</i> <i>G_OCH3</i>	CPTIMERW channel 3 output compare interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	



-	-
---	---

Example:

```
/* clear the CPTIMERW interrupt bit */
cptimerw_interrupt_flag_clear(CPTIMERW_INT_FLAG_UP);
```

## 3.8. 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 [3.8.1](#), the CRC firmware functions are introduced in chapter [3.8.2](#).

### 3.8.1. 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

### 3.8.2. 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 initializaion 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**

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

Example:

```
/* reset crc */
crc_deinit();
```

## crc\_data\_register\_reset

The description of crc\_data\_register\_reset is shown as below:

**Table 3-221. Function crc\_data\_register\_reset**

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

Example:

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

## crc\_reverse\_output\_data\_enable

The description of crc\_reverse\_output\_data\_enable is shown as below:

Table 3-222. 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 3-223. 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_input_data_reverse_config`

The description of `crc_input_data_reverse_config` is shown as below:

Table 3-224. Function `crc_input_data_reverse_config`

Function name	<code>crc_input_data_reverse_config</code>
---------------	--

<b>Function prototype</b>	void crc_input_data_reverse_config(uint32_t data_reverse);
<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
CRC_INPUT_DATA_NOT	input data is not reversed
CRC_INPUT_DATA_BYTE	input data is reversed on 8 bits
CRC_INPUT_DATA_HALFWORD	input data is reversed on 16 bits
CRC_INPUT_DATA_WORD	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\_data\_register\_read

The description of crc\_data\_register\_read is shown as below:

**Table 3-225. Function crc\_data\_register\_read**

<b>Function name</b>	crc_data_register_read
<b>Function prototype</b>	uint32_t crc_data_register_read(void);
<b>Function descriptions</b>	read the data register
<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>	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 3-226. Function crc\_free\_data\_register\_read**

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

Example:

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

uint8_t crc_value = 0;

crc_value = crc_free_data_register_read();
```

### crc\_free\_data\_register\_write

The description of crc\_free\_data\_register\_write is shown as below:

**Table 3-227. Function crc\_free\_data\_register\_write**

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

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 3-228. Function crc\_init\_data\_register\_write**

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

Example:

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

## 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>	crc_polynomial_size_set
<b>Function prototype</b>	void crc_polynomial_size_set(uint32_t poly_size);
<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
<i>CRC_CTL_PS_32</i>	32-bit polynomial for CRC calculation
<i>CRC_CTL_PS_16</i>	16-bit polynomial for CRC calculation
<i>CRC_CTL_PS_8</i>	8-bit polynomial for CRC calculation
<i>CRC_CTL_PS_7</i>	7-bit polynomial for CRC calculation
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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`**

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

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`**

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

Return value	
<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;

rcu_periph_clock_enable(RCU_CRC);

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>	calculate the CRC value of an data array
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>array</b>	pointer to 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
<code>INPUT_FORMAT_WORD</code>	input data in word format
<code>INPUT_FORMAT_HALFWORD</code>	input data in half-word format
<code>INPUT_FORMAT_BYTE</code>	input data in byte format
<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 array */

#define BUFFER_SIZE    6

uint32_t valcrc = 0;
```



```
static const uint32_t data_buffer[BUFFER_SIZE] = {  
  
    0x00001111, 0x00002222, 0x00003333, 0x00004444, 0x00005555, 0x00006666};  
  
    rcu_periph_clock_enable(RCU_CRC);  
  
    valcrc = crc_block_data_calculate(data_buffer, BUFFER_SIZE, INPUT_FORMAT_WORD);
```

## 3.9. 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 [3.9.1](#), the DAC firmware functions are introduced in chapter [3.9.2](#).

### 3.9.1. Descriptions 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_OUT1_R12DH	DACx_OUT1 12-bit right-aligned data holding register
DAC_OUT1_L12DH	DACx_OUT1 12-bit left-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
DAC_OUT0_DO	DACx_OUT0 data output register
DAC_OUT1_DO	DACx_OUT1 data output register

### 3.9.2. 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_sync_enable	enable DAC sync function
dac_sync_disable	disable DAC sync function
dac_connect_to_pin_enable	enable DAC connect to pin
dac_connect_to_pin_disable	disable DAC connect to pin
dac_connect_to_cmp_enable	enable DAC connect to CMP
dac_connect_to_cmp_disable	disable DAC connect to CMP
dac_output_value_get	get DAC output value
dac_data_set	set DAC data holding register value
dac_trigger_enable	enable DAC trigger
dac_trigger_disable	disable DAC trigger
dac_trigger_source_config	configure DAC trigger source

Function name	Function description
<code>dac_software_trigger_enable</code>	enable DAC software trigger
<code>dac_wave_mode_config</code>	configure DAC wave mode
<code>dac_lfsr_noise_config</code>	configure DAC LFSR noise mode
<code>dac_triangle_noise_config</code>	configure DAC triangle noise mode
<code>dac_concurrent_enable</code>	enable DAC concurrent mode
<code>dac_concurrent_disable</code>	disable DAC concurrent mode
<code>dac_concurrent_software_trigger_enable</code>	enable DAC concurrent software trigger
<code>dac_concurrent_data_set</code>	set DAC concurrent mode data holding register value

### **`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)
<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>	-
<b>Input parameter{in}</b>	
<b><code>dac_periph</code></b>	DAC peripheral

<i>DACx</i>	DAC peripheral selection(x = 0)
<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 */
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>	<code>dac_disable</code>
<b>Function prototype</b>	<code>void dac_disable(uint32_t dac_periph, uint8_t dac_out);</code>
<b>Function descriptions</b>	disable DAC
<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)
<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 */
dac_disable(DAC0, DAC_OUT0);
```

### **dac\_sync\_enable**

The description of `dac_sync_enable` is shown as below:

**Table 3-238. Function `dac_sync_enable`**

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

<b>Function prototype</b>	void dac_sync_enable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	enable DAC sync function
<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)
<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 sync function */
dac_sync_enable(DAC0, DAC_OUT0);
```

## **dac\_sync\_disable**

The description of dac\_sync\_disable is shown as below:

**Table 3-239. Function dac\_sync\_disable**

<b>Function name</b>	dac_sync_disable
<b>Function prototype</b>	void dac_sync_disable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	disable DAC sync function
<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)
<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 sync function */
dac_sync_disable(DAC0, DAC_OUT0);
```

## dac\_connect\_to\_pin\_enable

The description of dac\_connect\_to\_pin\_enable is shown as below:

**Table 3-240. Function dac\_connect\_to\_pin\_enable**

<b>Function name</b>	dac_connect_to_pin_enable
<b>Function prototype</b>	void dac_connect_to_pin_enable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	enable DAC connect to pin
<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)
<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 connect to pin */
```

```
dac_connect_to_pin_enable(DAC0, DAC_OUT0);
```

## dac\_connect\_to\_pin\_disable

The description of dac\_connect\_to\_pin\_disable is shown as below:

**Table 3-241. Function dac\_connect\_to\_pin\_disable**

<b>Function name</b>	dac_connect_to_pin_disable
<b>Function prototype</b>	void dac_connect_to_pin_disable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	disable DAC connect to pin
<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)
<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 connect to pin */
dac_connect_to_pin_disable(DAC0, DAC_OUT0);
```

### **dac\_connect\_to\_cmp\_enable**

The description of `dac_connect_to_cmp_enable` is shown as below:

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

Function name	<code>dac_connect_to_cmp_enable</code>
Function prototype	<code>void dac_connect_to_cmp_enable(uint32_t dac_periph, uint8_t dac_out);</code>
Function descriptions	enable DAC connect to CMP
Precondition	-
The called functions	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
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 connect to CMP */
dac_connect_to_cmp_enable(DAC0, DAC_OUT0);
```

### **dac\_connect\_to\_cmp\_disable**

The description of `dac_connect_to_cmp_disable` is shown as below:

**Table 3-243. Function `dac_connect_to_cmp_disable`**

Function name	<code>dac_connect_to_cmp_disable</code>
Function prototype	<code>void dac_connect_to_cmp_disable(uint32_t dac_periph, uint8_t dac_out);</code>
Function descriptions	disable DAC connect to CMP
Precondition	-
The called functions	-
Input parameter{in}	

<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
<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 connect to CMP */
```

```
dac_connect_to_cmp_disable(DAC0, DAC_OUT0);
```

### **dac\_output\_value\_get**

The description of `dac_output_value_get` is shown as below:

**Table 3-244. 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, uint8_t dac_out);</code>
<b>Function descriptions</b>	get DAC output 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)
<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>	
<b>uint16_t</b>	DAC output data (0~4095)

Example:

```
/* get the DAC0_OUT0 last data output value */
```

```
uint16_t data;
```

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

### **dac\_data\_set**

The description of `dac_data_set` is shown as below:



Table 3-245. 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
Precondition	-
The called functions	-
Input parameter{in}	
<code>dac_periph</code>	DAC peripheral
<code>DACx</code>	DAC peripheral selection(x = 0)
Input parameter{in}	
<code>dac_out</code>	DAC output
<code>DAC_OUTx</code>	DAC output channel selection(x = 0,1)
Input parameter{in}	
<code>dac_align</code>	DAC data alignment mode
<code>DAC_ALIGN_12B_R</code>	12-bit right-aligned data
<code>DAC_ALIGN_12B_L</code>	12-bit left-aligned data
Input parameter{in}	
<code>data</code>	data to be loaded (0~4095)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC0_OUT0 data holding register value */
```

```
dac_data_set(DAC0, DAC_OUT0, DAC_ALIGN_12B_R, 0xFFFF);
```

### `dac_trigger_enable`

The description of `dac_trigger_enable` is shown as below:

Table 3-246. Function `dac_trigger_enable`

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

Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### **dac\_trigger\_disable**

The description of dac\_trigger\_disable is shown as below:

**Table 3-247. Function dac\_trigger\_disable**

Function name	dac_trigger_disable
Function prototype	void dac_trigger_disable(uint32_t dac_periph, uint8_t dac_out);
Function descriptions	disable DAC trigger
Precondition	-
The called functions	-
Input parameter{in}	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
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 trigger */
dac_trigger_disable(DAC0, DAC_OUT0);
```

### **dac\_trigger\_source\_config**

The description of dac\_trigger\_source\_config is shown as below:

**Table 3-248. Function dac\_trigger\_source\_config**

Function name	dac_trigger_source_config
Function prototype	void dac_trigger_source_config(uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);
Function descriptions	configure DAC trigger source
Precondition	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
<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_EXTERNAL</i>	external trigger selected by EVIC
<i>DAC_TRIGGER_SOFTWARE</i>	software 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_EXTERNAL);
```

### **dac\_software\_trigger\_enable**

The description of `dac_software_trigger_enable` is shown as below:

**Table 3-249. 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)
<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 3-250. 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)
<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>	
-	-
<b>Return value</b>	
-	-

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 3-251. Function dac\_lfsr\_noise\_config**

<b>Function name</b>	dac_lfsr_noise_config
<b>Function prototype</b>	void dac_lfsr_noise_config(uint32_t dac_periph, uint8_t dac_out, uint32_t unmask_bits);

<b>Function descriptions</b>	configure DAC LFSR 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)
<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>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)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-252. Function `dac_triangle_noise_config`**

<b>Function name</b>	<code>dac_triangle_noise_config</code>
<b>Function prototype</b>	<code>void dac_triangle_noise_config(uint32_t dac_periph, uint8_t dac_out, uint32_t amplitude);</code>
<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)
<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_AMPLIT</i> <i>UDE_x</i>	$x = 2^n - 1$ (n = 1..12)
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure DAC0_OUT0 triangle noise mode */
```

```
dac_triangle_noise_config(DAC0, DAC_OUT0, DAC_TRIANGLE_AMPLITUDE_1);
```

### **dac\_concurrent\_enable**

The description of `dac_concurrent_enable` is shown as below:

**Table 3-253. Function `dac_concurrent_enable`**

<b>Function name</b>	<code>dac_concurrent_enable</code>
<b>Function prototype</b>	<code>void dac_concurrent_enable(uint32_t dac_periph);</code>
<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)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DAC0 concurrent mode */
```

```
dac_concurrent_enable(DAC0);
```

### **dac\_concurrent\_disable**

The description of `dac_concurrent_disable` is shown as below:

**Table 3-254. Function `dac_concurrent_disable`**

<b>Function name</b>	<code>dac_concurrent_disable</code>
<b>Function prototype</b>	<code>void dac_concurrent_disable(uint32_t dac_periph);</code>
<b>Function descriptions</b>	disable 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)
<b>Output parameter{out}</b>	

-	-
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 3-255. Function `dac_concurrent_software_trigger_enable`**

<b>Function name</b>	<code>dac_concurrent_software_trigger_enable</code>
<b>Function prototype</b>	<code>void dac_concurrent_software_trigger_enable(uint32_t dac_periph);</code>
<b>Function descriptions</b>	enable DAC concurrent 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)
<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 3-256. 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)

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

Example:

```
/* set DAC0 concurrent mode data holding register value */
```

```
dac_concurrent_data_set(DAC0, DAC_ALIGN_12B_R, 0xFFFF, 0xFFFF);
```

## 3.10. DBG

The DBG hold unit helps debugger to debug power saving mode. The DBG registers are listed in chapter [3.10.1](#). the DBG firmware functions are introduced in chapter [3.10.2](#).

### 3.10.1. Descriptions of Peripheral registers

DBG registers are listed in the table shown as below:

**Table 3-257. DBG Registers**

Registers	Descriptions
DBG_ID	DBG ID code register
DBG_CTL	DBG control register

### 3.10.2. Descriptions of Peripheral functions

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

**Table 3-258. DBG firmware function**

Function name	Function description
dbg_deinit	deinit DBG
dbg_id_get	read DBG_ID code register



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

### Enum dbg\_periph\_enum

**Table 3-259. Enum dbg\_periph\_enum**

Member name	Function description
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_TIMER0_HOLD	hold TIMER0 counter when core is halted
DBG_TIMER1_HOLD	hold TIMER1 counter when core is halted
DBG_TIMER2_HOLD	hold TIMER2 counter when core is halted
DBG_GPTIMER0_HOLD	hold GPTIMER0 counter when core is halted
DBG_CAN_HOLD	debug CAN kept when core is halted
DBG_I2C_HOLD	hold I2C smbus when core is halted
DBG_GPTIMER1_HOLD	hold GPTIMER1 counter when core is halted
DBG_CPTIMER0_HOLD	hold CPTIMER0 counter when core is halted
DBG_CPTIMER1_HOLD	hold CPTIMER1 counter when core is halted
DBG_TIMER7_HOLD	hold TIMER7 counter when core is halted
DBG_CAN_HOLD	debug CAN kept when core is halted
DBG_CPTIMERW_HOLD	hold CPTIMERW counter when core is halted

### dbg\_deinit

The description of dbg\_deinit is shown as below:

**Table 3-260. Function dbg\_deinit**

Function name	dbg_deinit
---------------	------------

<b>Function prototype</b>	void dbg_deinit(void);
<b>Function descriptions</b>	deinit DBG
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinit DBG*/
```

```
dbg_deinit();
```

### dbg\_id\_get

The description of dbg\_id\_get is shown as below:

**Table 3-261. Function dbg\_id\_get**

<b>Function name</b>	dbg_id_get
<b>Function prototype</b>	uint32_t dbg_id_get(void);
<b>Function descriptions</b>	Read DBG_ID code register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
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 3-262. Function dbg\_low\_power\_enable**

<b>Function name</b>	dbg_low_power_enable
<b>Function prototype</b>	void dbg_low_power_enable(uint32_t dbg_low_power);
<b>Function descriptions</b>	Enable 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>	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 3-263. Function `dbg_low_power_disable`

<b>Function name</b>	<code>dbg_low_power_disable</code>
<b>Function prototype</b>	<code>void dbg_low_power_disable(uint32_t dbg_low_power);</code>
<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
<code>DBG_LOW_POWER_SLEEP</code>	keep debugger connection during sleep mode
<code>DBG_LOW_POWER_DEEPSLEEP</code>	keep debugger connection during deepsleep mode
<code>DBG_LOW_POWER_STANDBY</code>	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\_periph\_enable**

The description of `dbg_periph_enable` is shown as below:

Table 3-264. Function `dbg_periph_enable`

<b>Function name</b>	<code>dbg_periph_enable</code>
<b>Function prototype</b>	<code>void dbg_periph_enable(dbg_periph_enum dbg_periph);</code>
<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 3-259. 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_TIMER0_HOLD);
```

### dbg\_periph\_disable

The description of dbg\_periph\_disable is shown as below:

**Table 3-265. 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 3-259. 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_TIMER0_HOLD);
```

### dbg\_trace\_pin\_enable

The description of dbg\_trace\_pin\_enable is shown as below:

Table 3-266. Function dbg\_trace\_pin\_enable

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

Example:

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

```
dbg_trace_pin_enable();
```

### dbg\_trace\_pin\_disable

The description of dbg\_trace\_pin\_disable is shown as below:

Table 3-267. Function dbg\_trace\_pin\_disable

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

Example:

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

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

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 [3.11.1](#), the DMAMUX firmware functions are introduced in chapter [3.11.2](#).

### 3.11.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

**Table 3-268. DMA Registers**

Registers	Descriptions
DMA_INTF	Interrupt flag register
DMA_INTC	Interrupt flag clear register
DMA_CHxCTL (x=0..5)	Channel x control register
DMA_CHxCNT (x=0..5)	Channel x counter register
DMA_CHxPADDR (x=0..5)	Channel x peripheral base address register
DMA_CHxMADDR (x=0..5)	Channel x memory base address register

DMAMUX registers are listed in the table shown as below:

**Table 3-269. DMAMUX Registers**

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

Registers	Descriptions
CFG (x=0..3)	
DMAMUX_RG_INT F	Request generator channel interrupt flag register
DMAMUX_RG_INT C	Rquest generator channel interrupt flag clear register

### 3.11.2. Descriptions of Peripheral functions

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

**Table 3-270. DMA firmware function**

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

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



Table 3-271. DMAMUX firmware function

Function name	Function description
dmamux_sync_struct_para_init	initialize the parameters of DMAMUX synchronization mode structure with the default values
dmamux_synchronization_init	initialize DMAMUX request multiplexer channel synchronization mode
dmamux_synchronization_enable	enable synchronization mode
dmamux_synchronization_disable	disable synchronization mode
dmamux_event_generation_enable	enable event generation
dmamux_event_generation_disable	disable event generation
dmamux_gen_struct_para_init	initialize the parameters of DMAMUX request generator structure with the default values
dmamux_request_generator_init	initialize DMAMUX request generator channel
dmamux_request_generator_channel_enable	enable DMAMUX request generator channel
dmamux_request_generator_channel_disable	disable DMAMUX request generator channel
dmamux_synchronization_polarity_config	configure synchronization input polarity
dmamux_request_forward_number_config	configure number of DMA requests to forward
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 3-272. 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 3-273. 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 3-274. 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 dmamux\_interrupt\_enum

**Table 3-275. Enum dmamux\_interrupt\_enum**

Member name	Function description
DMAMUX_INT_MU_XCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
DMAMUX_INT_MU_XCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
DMAMUX_INT_MU_XCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun interrupt
DMAMUX_INT_MU_XCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
DMAMUX_INT_MU_XCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun interrupt
DMAMUX_INT_MU_XCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun interrupt
DMAMUX_INT_MU_XCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun interrupt
DMAMUX_INT_MU_XCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun interrupt
DMAMUX_INT_MU_XCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun interrupt
DMAMUX_INT_MU_XCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun interrupt

DMAMUX_INT_MUXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun interrupt
DMAMUX_INT_MUXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun interrupt
DMAMUX_INT_GEN_NCH0_TO	DMAMUX request generator channel 0 trigger overrun interrupt
DMAMUX_INT_GEN_NCH1_TO	DMAMUX request generator channel 1 trigger overrun interrupt
DMAMUX_INT_GEN_NCH2_TO	DMAMUX request generator channel 2 trigger overrun interrupt
DMAMUX_INT_GEN_NCH3_TO	DMAMUX request generator channel 3 trigger overrun interrupt

### Enum dmamux\_flag\_enum

**Table 3-276. Enum dmamux\_flag\_enum**

Member name	Function description
DMAMUX_FLAG_MUXCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun flag
DMAMUX_FLAG_MUXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun flag
DMAMUX_FLAG_MUXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun flag
DMAMUX_FLAG_MUXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun flag
DMAMUX_FLAG_MUXCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun flag
DMAMUX_FLAG_MUXCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun flag
DMAMUX_FLAG_MUXCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun flag
DMAMUX_FLAG_MUXCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun flag
DMAMUX_FLAG_MUXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun flag
DMAMUX_FLAG_MUXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun flag
DMAMUX_FLAG_MUXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun flag
DMAMUX_FLAG_MUXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun flag
DMAMUX_FLAG_GEN_ENCH0_TO	DMAMUX request generator channel 0 trigger overrun flag
DMAMUX_FLAG_GEN_ENCH1_TO	DMAMUX request generator channel 1 trigger overrun flag

ENCH1_TO	
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 3-277. Enum dmamux\_interrupt\_flag\_enum**

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

## Enum dma\_channel\_enum

Table 3-278. 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

## Enum dmamux\_multiplexer\_channel\_enum

Table 3-279. Enum dmamux\_multiplexer\_channel\_enum

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

## Enum dmamux\_generator\_channel\_enum

Table 3-280. 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

## dma\_deinit

The description of dma\_deinit is shown as below:

**Table 3-281. 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(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a> :
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize DMA0 channel 0 registers*/
dma_deinit(DMA0, DMA_CH0);
```

## dma\_struct\_para\_init

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

**Table 3-282. Function dma\_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>	the initialization data needed to initialize DMA channel, refer to <a href="#">Table 3-272. Structure dma_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

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

## dma\_init

The description of dma\_init is shown as below:

**Table 3-283. Function dma\_init**

Function name	dma_init
Function prototype	void dma_init(uint32_t dma_periph, dma_channel_enum channelx, dma_parameter_struct init_struct);
Function descriptions	initialize DMA channel
Precondition	-
The called functions	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
Input parameter{in}	
<b>init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-272. Structure dma_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* DMA0 channel 0 initialize */
dma_parameter_struct dma_init_struct;

dma_struct_para_init(&dma_init_struct);
dma_init_struct.direction = DMA_PERIPHERAL_TO_MEMORY;
dma_init_struct.memory_addr = (uint32_t)g_destbuf;
dma_init_struct.memory_inc = DMA_MEMORY_INCREASE_ENABLE;
dma_init_struct.memory_width = DMA_MEMORY_WIDTH_8BIT;
dma_init_struct.number = TRANSFER_NUM;
dma_init_struct.periph_addr = (uint32_t)BANK0_WRITE_START_ADDR;
dma_init_struct.periph_inc = DMA_PERIPH_INCREASE_ENABLE;
```

```

dma_init_struct.periph_width = DMA_PERIPHERAL_WIDTH_8BIT;
dma_init_struct.priority = DMA_PRIORITY_ULTRA_HIGH;
dma_init(DMA0, DMA_CH0, dma_init_struct);

```

### dma\_circulation\_enable

The description of dma\_circulation\_enable is shown as below:

**Table 3-284. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* enable DMA0 channel 0 circulation mode */
dma_circulation_enable(DMA0, DMA_CH0);

```

### dma\_circulation\_disable

The description of dma\_circulation\_disable is shown as below:

**Table 3-285. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	



<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMA0 channel 0 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 3-286. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DMA0 channel 0 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 3-287. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-288. Function dma\_channel\_enable**

<b>Function name</b>	dma_channel_enable
<b>Function prototype</b>	void dma_channel_enable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	enable DMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DMA0 channel 0 */
dma_channel_enable(DMA0, DMA_CH0);
```

## dma\_channel\_disable

The description of dma\_channel\_disable is shown as below:

**Table 3-289. Function dma\_channel\_disable**

<b>Function name</b>	dma_channel_disable
<b>Function prototype</b>	void dma_channel_disable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	disable DMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMA0 channel 0 */
dma_channel_disable(DMA0, DMA_CH0);
```

## dma\_periph\_address\_config

The description of dma\_periph\_address\_config is shown as below:

**Table 3-290. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>address</b>	peripheral base address

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel 0 periph address */
#define BANK0_WRITE_START_ADDR          ((uint32_t)0x08004000)
dma_periph_address_config(DMA0, DMA_CH0, BANK0_WRITE_START_ADDR);
```

### **dma\_memory\_address\_config**

The description of dma\_memory\_address\_config is shown as below:

**Table 3-291. Function dma\_memory\_address\_config**

Function name	dma_memory_address_config
Function prototype	void dma_memory_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t address);
Function descriptions	set DMA memory base address
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
Input parameter{in}	
<b>address</b>	memory base address, 0 – 0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure DMA0 channel 0 memory address */
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 3-292. Function dma\_transfer\_number\_config**

Function name	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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>number</b>	data transfer number(0x0-0xFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DMA0 channel 0 transfer number */
#define TRANSFER_NUM          0x400
dma_transfer_number_config(DMA0, DMA_CH0, TRANSFER_NUM);
```

### **dma\_transfer\_number\_get**

The description of dma\_transfer\_number\_get is shown as below:

**Table 3-293. 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(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	DMA data transmission remaining quantity (0x0-0xFFFF)

Example:

```
/* get DMA0 channel 0 transfer number */
uint32_t number = 0;
number = dma_transfer_number_get(DMA0, DMA0, DMA_CH0);
```

### **dma\_priority\_config**

The description of dma\_priority\_config is shown as below:

**Table 3-294. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
DMAx(x=0, 1)	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
DMA_CHx(x=0..5)	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>priority</b>	priority Level of this channel
DMA_PRIORITY_LOW	low priority
DMA_PRIORITY_MEDIUM	medium priority
DMA_PRIORITY_HIGH	high priority
DMA_PRIORITY_ULTRA_HIGH	ultra high priority
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DMA0 channel 0 priority */
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 3-295. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<b>mwidth</b>	transfer data width of memory
<i>DMA_MEMORY_WIDT H_8BIT</i>	transfer data width of memory is 8-bit
<i>DMA_MEMORY_WIDT H_16BIT</i>	transfer data width of memory is 16-bit
<i>DMA_MEMORY_WIDT H_32BIT</i>	transfer data width of memory is 32-bit
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DMA0 channel 0 memory width */
dma_memory_width_config(DMA0, DMA_CH0, DMA_MEMORY_WIDTH_8BIT);
```

### **dma\_periph\_width\_config**

The description of dma\_periph\_width\_config is shown as below:

**Table 3-296. 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 width of peripheral
<b>Precondition</b>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel

<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. 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:

```
/* configure DMA0 channel 0 periph width */
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 3-297. Function dma\_memory\_increase\_enable**

<b>Function name</b>	dma_memory_increase_enable
<b>Function prototype</b>	void dma_memory_increase_enable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	enable next address increasement algorithm of memory
<b>Precondition</b>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DMA0 channel 0 memory increase */
dma_memory_increase_enable(DMA0, DMA_CH0);
```



## dma\_memory\_increase\_disable

The description of dma\_memory\_increase\_disable is shown as below:

**Table 3-298. Function dma\_memory\_increase\_disable**

<b>Function name</b>	dma_memory_increase_disable
<b>Function prototype</b>	void dma_memory_increase_disable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	disable next address increasement algorithm of memory
<b>Precondition</b>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMA0 channel 0 memory increase */
dma_memory_increase_disable(DMA0, DMA_CH0);
```

## dma\_periph\_increase\_enable

The description of dma\_periph\_increase\_enable is shown as below:

**Table 3-299. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* enable next address increasement algorithm of DMA0 channel 0 */
dma_periph_increase_enable(DMA0, DMA_CH0);
```

### **dma\_periph\_increase\_disable**

The description of dma\_periph\_increase\_disable is shown as below:

**Table 3-300. 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>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable next address increasement algorithm of DMA0 channel 0 */
dma_periph_increase_disable(DMA0, DMA_CH0);
```

### **dma\_transfer\_direction\_config**

The description of dma\_transfer\_direction\_config is shown as below:

**Table 3-301. Function dma\_transfer\_direction\_config**

<b>Function name</b>	dma_transfer_direction_config
<b>Function prototype</b>	void dma_transfer_direction_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t direction);
<b>Function descriptions</b>	configure the direction of data transfer on the channel
<b>Precondition</b>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
<b>Input parameter{in}</b>	
<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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DMA0 channel0 transfer direction */
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 3-302. Function dma\_flag\_get**

<b>Function name</b>	dma_flag_get
<b>Function prototype</b>	FlagStatus dma_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
<b>Function descriptions</b>	check DMA flag 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(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. 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>	

-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMA0 channel 0 flag */
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 3-303. 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 DMA a channel flag
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0,1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(x=0..5)	DMA channel selection, refer to <a href="#">Table 3-278. 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	
-	-

Example:

```
/* clear DMA0 channel 0 flag */
dma_flag_clear(DMA0, DMA_CH0, DMA_FLAG_FTF);
```

### dma\_interrupt\_enable

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

Table 3-304. Function dma\_interrupt\_enable

Function name	dma_interrupt_enable
Function prototype	void dma_interrupt_enable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t source);
Function descriptions	enable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0, 1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(x=0..5)	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
Input parameter{in}	
source	DMA interrupt source
DMA_INT_FTF	full transfer finish interrupt of channel
DMA_INT_HTF	half transfer finish interrupt of channel
DMA_INT_ERR	error interrupt of channel
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 channel 0 interrupt */
dma_interrupt_enable(DMA0, DMA_CH0, DMA_INT_FTF);
```

### dma\_interrupt\_disable

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

Table 3-305. Function dma\_interrupt\_disable

Function name	dma_interrupt_disable
Function prototype	void dma_interrupt_disable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t source);
Function descriptions	disable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx(x=0, 1)	DMA peripheral selection
Input parameter{in}	
channelx	DMA channel
DMA_CHx(x=0..5)	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>

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

Example:

```
/* disable DMA0 channel 0 interrupt */
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 3-306. 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>	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0,1)</i>	DMA peripheral selection
Input parameter{in}	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. Enum dma_channel_enum</a>
Input parameter{in}	
<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
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get DMA0 channel 3 interrupt flag */
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 3-307. 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 DMA a channel flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx(x=0, 1)</i>	DMA peripheral selection
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel
<i>DMA_CHx(x=0..5)</i>	DMA channel selection, refer to <a href="#">Table 3-278. 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:

```

/* clear DMA0 channel 3 interrupt flag */
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 3-308. 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 3-273. 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 3-309. Function dmamux\_synchronization\_init**

<b>Function name</b>	dmamux_synchronization_init
<b>Function prototype</b>	void dmamux_synchronization_init(dmamux_multiplexer_channel_enum channelx, dmamux_sync_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize DMAMUX request multiplexer channel synchronization mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..11)	DMAMUX channel selection, refer to <a href="#">Table 3-279. Enum dmamux_multiplexer_channel_enum</a>
<b>Input parameter{in}</b>	
<b>init_struct</b>	the initialization data needed to initialize DMAMUX request multiplexer channel synchronization mode, refer to <a href="#">Table 3-273. Structure dmamux_sync_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<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);
/* 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 3-310. Function dmamux\_synchronization\_enable**

Function name	dmamux_synchronization_enable
Function prototype	void dmamux_synchronization_enable(dmamux_multiplexer_channel_enum channelx);
Function descriptions	enable synchronization mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..11)	DMAMUX channel selection, refer to <a href="#">Table 3-279. 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 3-311. 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}	

<b>channelx</b>	specify which DMAMUX request multiplexer channel is initialized
<i>DMAMUX_MUXCHx</i> (x=0..11)	DMAMUX channel selection, refer to <a href="#">Table 3-279. Enum <i>dmamux_multiplexer_channel_enum</i></a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-312. Function dmamux\_event\_generation\_enable**

<b>Function name</b>	dmamux_event_generation_enable
<b>Function prototype</b>	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>	specify which DMAMUX request multiplexer channel is initialized
<i>DMAMUX_MUXCHx</i> (x=0..11)	DMAMUX channel selection, refer to <a href="#">Table 3-279. Enum <i>dmamux_multiplexer_channel_enum</i></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 3-313. 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);

	channelx);
<b>Function descriptions</b>	disable event generation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	specify which DMAMUX request multiplexer channel is initialized
<b>DMAMUX_MUXCHx(x=0..11)</b>	DMAMUX channel selection, refer to <a href="#">Table 3-279. 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 3-314. Function dmamux\_gen\_struct\_para\_init**

<b>Function name</b>	dmamux_gen_struct_para_init
<b>Function prototype</b>	void dmamux_gen_struct_para_init(dmamux_gen_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize the parameters of DMAMUX request generator 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 generator channel, refer to <a href="#">Table 3-274. Structure dmamux_gen_parameter_struct</a>
<b>Return value</b>	
-	-

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 3-315. 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}	
<b>channelx</b>	specify which DMAMUX request generator channel is initialized
<code>DMAMUX_GENCHx(x=0..3)</code>	DMAMUX generation channel selection, refer to <a href="#">Table 3-280. Enum <code>dmamux_generator_channel_enum</code></a>
Input parameter{in}	
<b>init_struct</b>	the initialization data needed to initialize DMAMUX request generator channel, refer to <a href="#">Table 3-274. 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 3-316. Function `dmamux_request_generator_channel_enable`

Function name	<code>dmamux_request_generator_channel_enable</code>
Function prototype	<code>void dmamux_request_generator_channel_enable(dmamux_generator_channel_enum channelx);</code>
Function descriptions	enable DMAMUX request generator channel
Precondition	-
The called functions	-
Input parameter{in}	
<b>channelx</b>	specify which DMAMUX request generator channel is initialized
<code>DMAMUX_GENCHx(x=0..3)</code>	DMAMUX generation channel selection, refer to <a href="#">Table 3-280. Enum <code>dmamux_generator_channel_enum</code></a>

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMAMUX request generator channel */
dmamux_request_generator_channel_enable(DMAMUX_GENCH0);
```

### dmamux\_request\_generator\_channel\_disable

The description of dmamux\_request\_generator\_channel\_disable is shown as below:

**Table 3-317. Function dmamux\_request\_generator\_channel\_disable**

Function name	dmamux_request_generator_channel_disable
Function prototype	void dmamux_request_generator_channel_disable(dmamux_generator_channel_enum channelx);
Function descriptions	disable DMAMUX request generator channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
DMAMUX_GENCHx(x=0..3)	DMAMUX generation channel selection, refer to <a href="#">Table 3-280. Enum dmamux_generator_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMAMUX request generator channel */
dmamux_request_generator_channel_disable(DMAMUX_GENCH0);
```

### dmamux\_synchronization\_polarity\_config

The description of dmamux\_synchronization\_polarity\_config is shown as below:

**Table 3-318. Function dmamux\_synchronization\_polarity\_config**

Function name	dmamux_synchronization_polarity_config
Function prototype	void dmamux_synchronization_polarity_config(dmamux_multiplexer_channel_enum channelx, uint32_t polarity);
Function descriptions	configure synchronization input polarity
Precondition	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	specify which DMAMUX request multiplexer channel is initialized
<i>DMAMUX_MUXCHx(x=0..11)</i>	DMAMUX channel selection, refer to <a href="#">Table 3-279. Enum <i>dmamux_multiplexer_channel_enum</i></a>
<b>Input parameter{in}</b>	
<b>polarity</b>	synchronization input polarity
<i>DMAMUX_SYNC_NO_EVENT</i>	no event detection
<i>DMAMUX_SYNC_RISING</i>	rising edge
<i>DMAMUX_SYNC_FALLING</i>	falling edge
<i>DMAMUX_SYNC_RISING_FALLING</i>	rising and falling edges
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-319. Function dmamux\_request\_forward\_number\_config**

<b>Function name</b>	dmamux_request_forward_number_config
<b>Function prototype</b>	void dmamux_request_forward_number_config(dmamux_multiplexer_channel_enum channelx, uint32_t number);
<b>Function descriptions</b>	configure number of DMA requests to forward
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	specify which DMAMUX request multiplexer channel is initialized
<i>DMAMUX_MUXCHx(x=0..11)</i>	DMAMUX channel selection, refer to <a href="#">Table 3-279. Enum <i>dmamux_multiplexer_channel_enum</i></a>
<b>Input parameter{in}</b>	
<b>number</b>	DMA requests number to forward (1 - 32)
<b>Output parameter{out}</b>	
-	-

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 3-320. Function dmamux\_sync\_id\_config**

Function name	dmamux_sync_id_config
Function prototype	void dmamux_sync_id_config(dmamux_multiplexer_channel_enum channelx, uint32_t id);
Function descriptions	configure synchronization input identification
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request multiplexer channel is initialized
DMAMUX_MUXCHx(x=0..11)	DMAMUX channel selection, refer to <a href="#">Table 3-279. Enum dmamux_multiplexer_channel_enum</a>
Input parameter{in}	
id	synchronization input identification
DMAMUX_SYNC_EXTI0	synchronization input is EXTI0
DMAMUX_SYNC_EXTI1	synchronization input is EXTI1
DMAMUX_SYNC_EXTI2	synchronization input is EXTI2
DMAMUX_SYNC_EXTI3	synchronization input is EXTI3
DMAMUX_SYNC_EXTI4	synchronization input is EXTI4
DMAMUX_SYNC_EXTI5	synchronization input is EXTI5
DMAMUX_SYNC_EXTI6	synchronization input is EXTI6
DMAMUX_SYNC_EXTI7	synchronization input is EXTI7
DMAMUX_SYNC_EXTI8	synchronization input is EXTI8
DMAMUX_SYNC_EXTI9	synchronization input is EXTI9

<i>DMAMUX_SYNC_EXTI</i> 10	synchronization input is EXTI10
<i>DMAMUX_SYNC_EXTI</i> 11	synchronization input is EXTI11
<i>DMAMUX_SYNC_EXTI</i> 12	synchronization input is EXTI12
<i>DMAMUX_SYNC_EXTI</i> 13	synchronization input is EXTI13
<i>DMAMUX_SYNC_EXTI</i> 14	synchronization input is EXTI14
<i>DMAMUX_SYNC_EXTI</i> 15	synchronization input is EXTI15
<i>DMAMUX_SYNC_EVT</i> <i>X_OUT0</i>	synchronization input is Evt_out0
<i>DMAMUX_SYNC_EVT</i> <i>X_OUT1</i>	synchronization input is Evt_out1
<i>DMAMUX_SYNC_EVT</i> <i>X_OUT2</i>	synchronization input is Evt_out2
<i>DMAMUX_SYNC_EVT</i> <i>X_OUT3</i>	synchronization input is Evt_out3
<i>DMAMUX_SYNC_TIM</i> <i>ER20_CH0_O</i>	synchronization input is LPTIMER_OUT
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-321. 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>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	specify which DMAMUX request multiplexer channel is initialized



DMAMUX_MUXCHx(x=0..11)	DMAMUX channel selection, refer to <a href="#">Table 3-279. Enum <i>dmamux_multiplexer_channel_enum</i></a>
<b>Input parameter{in}</b>	
<b>id</b>	input DMA request identification
DMA_REQUEST_M2M	内存到内存传输
DMA_REQUEST_GENERATOR0	DMAMUX GENERATOR0 request
DMAMUX_REQUEST_GENERATOR1	DMAMUX GENERATOR1 request
DMAMUX_REQUEST_GENERATOR2	DMAMUX GENERATOR2 request
DMA_REQUEST_GENERATOR3	DMAMUX GENERATOR3 request
DMA_REQUEST_ADC0_A	DMAMUX ADC0_A request
DMA_REQUEST_ADC2_A	DMAMUX ADC2_A request
DMA_REQUEST_ADC0_B	DMAMUX ADC0_B request
DMA_REQUEST_ADC2_B	DMAMUX ADC2_B request
DMA_REQUEST_ADC0_C	DMAMUX ADC0_C request
DMA_REQUEST_ADC2_C	DMAMUX ADC2_C request
DMA_REQUEST_I2C_RX	DMAMUX I2C_RX request
DMA_REQUEST_I2C_TX	DMAMUX I2C_TX request
DMA_REQUEST_UART0_RX	DMAMUX UART0_RX request
DMA_REQUEST_UART0_TX	DMAMUX UART0_TX request
DMA_REQUEST_UART1_RX	DMAMUX UART1_RX request
DMA_REQUEST_UART1_TX	DMAMUX UART1_TX request
DMA_REQUEST_UART2_RX	DMAMUX UART2_RX request
DMA_REQUEST_UART2_TX	DMAMUX UART2_TX request
DMA_REQUEST_UART3_RX	DMAMUX UART3_RX request

<i>DMA_REQUEST_UAR T3_TX</i>	DMAMUX UART3_TX request
<i>DMA_REQUEST_SPI0 _RX</i>	DMAMUX SPI0_RX request
<i>DMA_REQUEST_SPI0 _TX</i>	DMAMUX SPI0_TX request
<i>DMA_REQUEST_TIME R0_UP</i>	DMAMUX TIMER0_UP request
<i>DMA_REQUEST_TIME R0_CO</i>	DMAMUX TIMER0_CO request
<i>DMA_REQUEST_TIME R0_TI</i>	DMAMUX TIMER0_TI request
<i>DMA_REQUEST_TIME R0_CH0</i>	DMAMUX TIMER0_CH0 request
<i>DMA_REQUEST_TIME R0_CH1</i>	DMAMUX TIMER0_CH1 request
<i>DMA_REQUEST_TIME R0_CH2</i>	DMAMUX TIMER0_CH2 request
<i>DMA_REQUEST_TIME R0_CH3</i>	DMAMUX TIMER0_CH3 request
<i>DMA_REQUEST_TIME R0_MCH0</i>	DMAMUX TIMER0_MCH0 request
<i>DMA_REQUEST_TIME R0_MCH1</i>	DMAMUX TIMER0_MCH1 request
<i>DMA_REQUEST_TIME R0_MCH2</i>	DMAMUX TIMER0_MCH2 request
<i>DMA_REQUEST_TIME R0_MCH3</i>	DMAMUX TIMER0_MCH3 request
<i>DMA_REQUEST_TIME R7_UP</i>	DMAMUX TIMER7_UP request
<i>DMA_REQUEST_TIME R7_CO</i>	DMAMUX TIMER7_CO request
<i>DMA_REQUEST_TIME R7_TI</i>	DMAMUX TIMER7_TI request
<i>DMA_REQUEST_TIME R7_CH0</i>	DMAMUX TIMER7_CH0 request
<i>DMA_REQUEST_TIME R7_CH1</i>	DMAMUX TIMER7_CH1 request
<i>DMA_REQUEST_TIME R7_CH2</i>	DMAMUX TIMER7_CH2 request
<i>DMA_REQUEST_TIME R7_CH3</i>	DMAMUX TIMER7_CH3 request
<i>DMA_REQUEST_TIME</i>	DMAMUX TIMER7_MCH0 request

<i>R7_MCH0</i>	
<i>DMA_REQUEST_TIME</i> <i>R7_MCH1</i>	DMAMUX TIMER7_MCH1 request
<i>DMA_REQUEST_TIME</i> <i>R7_MCH2</i>	DMAMUX TIMER7_MCH2 request
<i>DMA_REQUEST_TIME</i> <i>R7_MCH3</i>	DMAMUX TIMER7_MCH3 request
<i>DMA_REQUEST_TIME</i> <i>R1_UP</i>	DMAMUX TIMER1_UP request
<i>DMA_REQUEST_TIME</i> <i>R1_TI</i>	DMAMUX TIMER1_TI request
<i>DMA_REQUEST_TIME</i> <i>R1_CH0</i>	DMAMUX TIMER1_CH0 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH1</i>	DMAMUX TIMER1_CH1 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH2</i>	DMAMUX TIMER1_CH2 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH3</i>	DMAMUX TIMER1_CH3 request
<i>DMA_REQUEST_TIME</i> <i>R2_UP</i>	DMAMUX TIMER2_UP request
<i>DMA_REQUEST_TIME</i> <i>R2_TI</i>	DMAMUX TIMER2_TI request
<i>DMA_REQUEST_TIME</i> <i>R2_CH0</i>	DMAMUX TIMER2_CH0 request
<i>DMA_REQUEST_TIME</i> <i>R2_CH1</i>	DMAMUX TIMER2_CH1 request
<i>DMA_REQUEST_TIME</i> <i>R2_CH2</i>	DMAMUX TIMER2_CH2 request
<i>DMA_REQUEST_TIME</i> <i>R2_CH3</i>	DMAMUX TIMER2_CH3 request
<i>DMA_REQUEST_TMU</i> <i>_TX</i>	DMAMUX TMU_TX request
<i>DMA_REQUEST_TMU</i> <i>_RX</i>	DMAMUX TMU_RX request
<i>DMA_REQUEST_GPTI</i> <i>MER0_UP</i>	DMAMUX GPTIMER0_UP request
<i>DMA_REQUEST_GPTI</i> <i>MER0_PRD</i>	DMAMUX GPTIMER0_PRD request
<i>DMA_REQUEST_GPTI</i> <i>MER0_CH0</i>	DMAMUX GPTIMER0_CH0 request
<i>DMA_REQUEST_GPTI</i> <i>MER0_CH1</i>	DMAMUX GPTIMER0_CH1 request

<i>DMA_REQUEST_GPTIMER1_UP</i>	DMAMUX GPTIMER1_UP request
<i>DMA_REQUEST_GPTIMER1_PRD</i>	DMAMUX GPTIMER1_PRD request
<i>DMA_REQUEST_GPTIMER1_CH0</i>	DMAMUX GPTIMER1_CH0 request
<i>DMA_REQUEST_GPTIMER1_CH1</i>	DMAMUX GPTIMER1_CH1 request
<i>DMA_REQUEST_EVIC0</i>	DMAMUX EVIC0 request
<i>DMA_REQUEST_EVIC1</i>	DMAMUX EVIC1 request
<i>DMA_REQUEST_EVIC2</i>	DMAMUX EVIC2 request
<i>DMA_REQUEST_EVIC3</i>	DMAMUX EVIC3 request
<i>DMA_REQUEST_EVIC4</i>	DMAMUX EVIC4 request
<i>DMA_REQUEST_EVIC5</i>	DMAMUX EVIC5 request
<i>DMA_REQUEST_EVIC6</i>	DMAMUX EVIC6 request
<i>DMA_REQUEST_EVIC7</i>	DMAMUX EVIC7 request
<i>DMA_REQUEST_EVIC8</i>	DMAMUX EVIC8 request
<i>DMA_REQUEST_EVIC9</i>	DMAMUX EVIC9 request
<i>DMA_REQUEST_ADC0_D</i>	DMAMUX ADC0_D request
<i>DMA_REQUEST_ADC2_D</i>	DMAMUX ADC2_D request
<i>DMA_REQUEST_GPTIMER0_CH0COMV_ADD</i>	DMAMUX GPTIMER0_CH0COMV_ADD request
<i>DMA_REQUEST_GPTIMER0_CH1COMV_ADD</i>	DMAMUX GPTIMER0_CH1COMV_ADD request
<i>DMA_REQUEST_GPTIMER1_CH0COMV_ADD</i>	DMAMUX GPTIMER1_CH0COMV_ADD request
<i>DMA_REQUEST_GPTIMER1_CH1COMV_ADD</i>	DMAMUX GPTIMER1_CH1COMV_ADD request

<i>DMA_REQUEST_EVIC</i> 10	DMAMUX EVIC10 request
<i>DMA_REQUEST_EVIC</i> 11	DMAMUX EVIC11 request
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-322. Function dmamux\_trigger\_polarity\_config**

<b>Function name</b>	dmamux_trigger_polarity_config
<b>Function prototype</b>	void dmamux_trigger_polarity_config(dmamux_generator_channel_enum channelx, uint32_t polarity);
<b>Function descriptions</b>	configure trigger input polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	specify which DMAMUX request generator channel is initialized
<i>DMAMUX_GENCHx</i> (x=0..3)	DMAMUX generation channel selection, refer to <a href="#">Table 3-280. Enum dmamux_generator_channel_enum</a>
<b>Input parameter{in}</b>	
<b>polarity</b>	trigger input polarity
<i>DMAMUX_GEN_NO_EVENT</i>	no event detection
<i>DMAMUX_GEN_RISING</i>	rising edge
<i>DMAMUX_GEN_FALLING</i>	falling edge
<i>DMAMUX_GEN_RISING_FALLING</i>	rising and falling edges
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-323. Function dmamux\_request\_generate\_number\_config**

Function name	dmamux_request_generate_number_config
Function prototype	void dmamux_request_generate_number_config(dmamux_generator_channel_enum channelx, uint32_t number);
Function descriptions	configure number of DMA requests to be generated
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
DMAMUX_GENCHx(x=0..3)	DMAMUX generation channel selection, refer to <a href="#">Table 3-280. 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 3-324. Function dmamux\_trigger\_id\_config**

Function name	dmamux_trigger_id_config
Function prototype	void dmamux_trigger_id_config(dmamux_generator_channel_enum channelx, uint32_t id);
Function descriptions	configure trigger input identification
Precondition	-
The called functions	-
Input parameter{in}	
channelx	specify which DMAMUX request generator channel is initialized
DMAMUX_GENCHx(x=0..3)	DMAMUX generation channel selection, refer to <a href="#">Table 3-280. Enum dmamux_generator_channel_enum</a>

Input parameter{in}	
<b>id</b>	trigger input identification
<i>DMAMUX_TRIGGER_EXTI0</i>	trigger input is EXTI0
<i>DMAMUX_TRIGGER_EXTI1</i>	trigger input is EXTI1
<i>DMAMUX_TRIGGER_EXTI2</i>	trigger input is EXTI2
<i>DMAMUX_TRIGGER_EXTI3</i>	trigger input is EXTI3
<i>DMAMUX_TRIGGER_EXTI4</i>	trigger input is EXTI4
<i>DMAMUX_TRIGGER_EXTI5</i>	trigger input is EXTI5
<i>DMAMUX_TRIGGER_EXTI6</i>	trigger input is EXTI6
<i>DMAMUX_TRIGGER_EXTI7</i>	trigger input is EXTI7
<i>DMAMUX_TRIGGER_EXTI8</i>	trigger input is EXTI8
<i>DMAMUX_TRIGGER_EXTI9</i>	trigger input is EXTI9
<i>DMAMUX_TRIGGER_EXTI10</i>	trigger input is EXTI10
<i>DMAMUX_TRIGGER_EXTI11</i>	trigger input is EXTI11
<i>DMAMUX_TRIGGER_EXTI12</i>	trigger input is EXTI12
<i>DMAMUX_TRIGGER_EXTI13</i>	trigger input is EXTI13
<i>DMAMUX_TRIGGER_EXTI14</i>	trigger input is EXTI14
<i>DMAMUX_TRIGGER_EXTI15</i>	trigger input is EXTI15
<i>DMAMUX_TRIGGER_EVT_OUT0</i>	trigger input is Evt_out0
<i>DMAMUX_TRIGGER_EVT_OUT1</i>	trigger input is Evt_out1
<i>DMAMUX_TRIGGER_EVT_OUT2</i>	trigger input is Evt_out2
<i>DMAMUX_TRIGGER_EVT_OUT3</i>	trigger input is Evt_out3
Output parameter{out}	

-	-
Return value	
-	-

Example:

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

### dmamux\_flag\_get

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

**Table 3-325. Function dmamux\_flag\_get**

Function name	dmamux_flag_get
Function prototype	FlagStatus dmamux_flag_get(dmamux_flag_enum flag);
Function descriptions	get DMAMUX flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	flag type, refer to <a href="#">Table 3-276. Enum dmamux_flag_enum</a>
DMAMUX_FLAG_MUX CH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun flag
DMAMUX_FLAG_MUX CH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun flag
DMAMUX_FLAG_MUX CH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun flag
DMAMUX_FLAG_MUX CH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun flag
DMAMUX_FLAG_MUX CH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun flag
DMAMUX_FLAG_MUX CH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun flag
DMAMUX_FLAG_MUX CH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun flag
DMAMUX_FLAG_MUX CH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun flag
DMAMUX_FLAG_MUX CH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun flag
DMAMUX_FLAG_MUX CH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun flag
DMAMUX_FLAG_MUX CH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun flag
DMAMUX_FLAG_MUX CH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun flag



<i>DMAMUX_FLAG_GEN</i> <i>CH0_TO</i>	DMAMUX request generator channel 0 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH1_TO</i>	DMAMUX request generator channel 1 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH2_TO</i>	DMAMUX request generator channel 2 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH3_TO</i>	DMAMUX request generator channel 3 trigger overrun flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

### dmamux\_flag\_clear

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

**Table 3-326. Function dmamux\_flag\_clear**

<b>Function name</b>	dmamux_flag_clear
<b>Function prototype</b>	void dmamux_flag_clear(dmamux_flag_enum flag);
<b>Function descriptions</b>	clear DMAMUX flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	flag type, refer to <a href="#">Table 3-276. Enum dmamux_flag_enum</a>
<i>DMAMUX_FLAG_MUX</i> <i>CH0_SO</i>	DMAMUX request multiplexer channel 0 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH1_SO</i>	DMAMUX request multiplexer channel 1 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH2_SO</i>	DMAMUX request multiplexer channel 2 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH3_SO</i>	DMAMUX request multiplexer channel 3 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH4_SO</i>	DMAMUX request multiplexer channel 4 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH5_SO</i>	DMAMUX request multiplexer channel 5 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH6_SO</i>	DMAMUX request multiplexer channel 6 synchronization overrun flag

<i>DMAMUX_FLAG_MUX</i> <i>CH7_SO</i>	DMAMUX request multiplexer channel 7 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH8_SO</i>	DMAMUX request multiplexer channel 8 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH9_SO</i>	DMAMUX request multiplexer channel 9 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH10_SO</i>	DMAMUX request multiplexer channel 10 synchronization overrun flag
<i>DMAMUX_FLAG_MUX</i> <i>CH11_SO</i>	DMAMUX request multiplexer channel 11 synchronization overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH0_TO</i>	DMAMUX request generator channel 0 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH1_TO</i>	DMAMUX request generator channel 1 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH2_TO</i>	DMAMUX request generator channel 2 trigger overrun flag
<i>DMAMUX_FLAG_GEN</i> <i>CH3_TO</i>	DMAMUX request generator channel 3 trigger overrun flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-327. Function dmamux\_interrupt\_enable**

<b>Function name</b>	dmamux_interrupt_enable
<b>Function prototype</b>	void dmamux_interrupt_enable(dmamux_interrupt_enum interrupt);
<b>Function descriptions</b>	enable DMAMUX interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	specify which interrupt to enable
<i>DMAMUX_INT_MUXC</i> <i>H0_SO</i>	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC</i> <i>H1_SO</i>	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC</i> <i>H2_SO</i>	DMAMUX request multiplexer channel 2 synchronization overrun interrupt

<i>DMAMUX_INT_MUXC H3_SO</i>	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H4_SO</i>	DMAMUX request multiplexer channel 4 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H5_SO</i>	DMAMUX request multiplexer channel 5 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H6_SO</i>	DMAMUX request multiplexer channel 6 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H7_SO</i>	DMAMUX request multiplexer channel 7 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H8_SO</i>	DMAMUX request multiplexer channel 8 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H9_SO</i>	DMAMUX request multiplexer channel 9 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H10_SO</i>	DMAMUX request multiplexer channel 10 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H11_SO</i>	DMAMUX request multiplexer channel 11 synchronization overrun interrupt
<i>DMAMUX_INT_GENC H0_TO</i>	DMAMUX request generator channel 0 trigger overrun interrupt
<i>DMAMUX_INT_GENC H1_TO</i>	DMAMUX request generator channel 1 trigger overrun interrupt
<i>DMAMUX_INT_GENC H2_TO</i>	DMAMUX request generator channel 2 trigger overrun interrupt
<i>DMAMUX_INT_GENC H3_TO</i>	DMAMUX request generator channel 3 trigger overrun interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-328. Function dmamux\_interrupt\_disable**

<b>Function name</b>	dmamux_interrupt_disable
<b>Function prototype</b>	void dmamux_interrupt_disable(dmamux_interrupt_enum interrupt);
<b>Function descriptions</b>	disable DMAMUX interrupt
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
interrupt	specify which interrupt to disable
<i>DMAMUX_INT_MUXC H0_SO</i>	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H1_SO</i>	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H2_SO</i>	DMAMUX request multiplexer channel 2 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H3_SO</i>	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H4_SO</i>	DMAMUX request multiplexer channel 4 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H5_SO</i>	DMAMUX request multiplexer channel 5 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H6_SO</i>	DMAMUX request multiplexer channel 6 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H7_SO</i>	DMAMUX request multiplexer channel 7 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H8_SO</i>	DMAMUX request multiplexer channel 8 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H9_SO</i>	DMAMUX request multiplexer channel 9 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H10_SO</i>	DMAMUX request multiplexer channel 10 synchronization overrun interrupt
<i>DMAMUX_INT_MUXC H11_SO</i>	DMAMUX request multiplexer channel 11 synchronization overrun interrupt
<i>DMAMUX_INT_GENC H0_TO</i>	DMAMUX request generator channel 0 trigger overrun interrupt
<i>DMAMUX_INT_GENC H1_TO</i>	DMAMUX request generator channel 1 trigger overrun interrupt
<i>DMAMUX_INT_GENC H2_TO</i>	DMAMUX request generator channel 2 trigger overrun interrupt
<i>DMAMUX_INT_GENC H3_TO</i>	DMAMUX request generator channel 3 trigger overrun interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

## dmamux\_interrupt\_flag\_get

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

**Table 3-329. 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 3-277. Enum dmamux_interrupt_flag_enum</a>
DMAMUX_INT_FLAG_MUXCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_MUXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun interrupt flag
DMAMUX_INT_FLAG_GENCH0_TO	DMAMUX request generator channel 0 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH1_TO	DMAMUX request generator channel 1 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH2_TO	DMAMUX request generator channel 2 trigger overrun interrupt flag
DMAMUX_INT_FLAG_GENCH3_TO	DMAMUX request generator channel 3 trigger overrun interrupt flag

GENCH3_TO	
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

### dmamux\_interrupt\_flag\_clear

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

**Table 3-330. Function dmamux\_interrupt\_flag\_clear**

<b>Function name</b>	dmamux_interrupt_flag_clear
<b>Function prototype</b>	FlagStatus dmamux_interrupt_flag_get(dmamux_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	clear DMAMUX interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>int_flag</b>	flag type, refer to <a href="#">Table 3-277. Enum dmamux_interrupt_flag_enum</a>
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

<i>MUXCH9_SO</i>	flag
<i>DMAMUX_INT_FLAG_MUXCH10_SO</i>	DMAMUX request multiplexer channel 10 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_MUXCH11_SO</i>	DMAMUX request multiplexer channel 11 synchronization overrun interrupt flag
<i>DMAMUX_INT_FLAG_GENCH0_TO</i>	DMAMUX request generator channel 0 trigger overrun interrupt flag
<i>DMAMUX_INT_FLAG_GENCH1_TO</i>	DMAMUX request generator channel 1 trigger overrun interrupt flag
<i>DMAMUX_INT_FLAG_GENCH2_TO</i>	DMAMUX request generator channel 2 trigger overrun interrupt flag
<i>DMAMUX_INT_FLAG_GENCH3_TO</i>	DMAMUX request generator channel 3 trigger overrun interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

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

```

## 3.12. EVIC

The event interconnection unit (EVIC) allows software to select event signals generated by various peripheral modules for a variety of applications. EVIC provides a flexible mechanism for a peripheral to select different event source. The EVIC registers are listed in chapter [3.12.1](#), the EVIC firmware functions are introduced in chapter [3.12.2](#).

### 3.12.1. Descriptions of Peripheral registers

EVIC registers are listed in the table shown as below:

**Table 3-331. EVIC registers**

Registers	Descriptions
EVIC_SWEV	Software event register
EVIC_SGIOx	Event interconnect for single I/O register x
EVIC INGRPE	Event interconnect for input group E register
EVIC INGRPF	Event interconnect for input group F register
EVIC_OUTGRPE	Event interconnect for output group E register
EVIC_OUTGRPF	Event interconnect for output group F register

Registers	Descriptions
EVIC_DAC0COV	Event interconnect for DAC0 conversion register
EVIC_ADC0COV	Event interconnect for ADC0 conversion register
EVIC_ADC2COV	Event interconnect for ADC2 conversion register
EVIC_RCU	Event interconnect for RCU register
EVIC_TIMER0	Event interconnect for TIMER0 register
EVIC_TIMER7	Event interconnect for TIMER7 register
EVIC_CPTIMER0	Event interconnect for CPTIMER0 register
EVIC_CPTIMERW	Event interconnect for CPTIMERW register
EVIC_TIMER1	Event interconnect for TIMER1 register
EVIC_TIMER2	Event interconnect for TIMER2 register
EVIC_GPTIMER_0	Event interconnect for GPTIMER register 0
EVIC_GPTIMER_1	Event interconnect for GPTIMER register 1
EVIC_GPTIMER_2	Event interconnect for GPTIMER register 2
EVIC_GPTIMER_3	Event interconnect for GPTIMER register 3
EVIC_DAC0EN	Event interconnect for DAC0 enable register
EVIC_DMAMUX_0	Event interconnect for DMAMUX register 0
EVIC_DMAMUX_1	Event interconnect for DMAMUX register 1
EVIC_DMAMUX_2	Event interconnect for DMAMUX register 2
EVIC_DMAMUX_3	Event interconnect for DMAMUX register 3
EVIC_DMAMUX_4	Event interconnect for DMAMUX register 4
EVIC_DMAMUX_5	Event interconnect for DMAMUX register 5
EVIC_SMCFG0	Slave mode configuration register 0
EVIC_SMCFG1	Slave mode configuration register 1
EVIC_SGIOCFGx	Single I/O configuration register x
EVIC_GRPFCFG	Group E configuration register
EVIC_GRPEDH	Group E data holding register
EVIC_GRPFCFG	Group F configuration register
EVIC_GRPFDH	Group F data holding register

### 3.12.2. Descriptions of Peripheral functions

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

**Table 3-332. EVIC firmware function**

Function name	Function description
evic_init	set the event source for target peripheral
evic_event_source_get	get the trigger input signal for target peripheral
evic_register_lock_set	lock the event interconnect register
evic_register_lock_get	get the event interconnect register lock status
evic_cptimer_slave_mode_select	select CPTIMERW or CPTIMER0 slave mode
evic_group_member_config	configure I/O group member
evic_group_edge_detection_config	configure I/O group input detection edge



Function name	Function description
evic_group_output_level_config	configure I/O group output level
evic_group_overwrite_enable	enable I/O group overwrite function
evic_group_overwrite_disable	disable I/O group overwrite function
evic_data_set	set I/O group holding data value
evic_data_get	get I/O group holding data value
evic_single_io_config	configure evic single I/O
evic_single_io_edge_detection_config	configure evic single I/O input detection edge
evic_single_io_output_level_config	configure evic single I/O output level
evic_register_write_enable	enable software event register write
evic_register_write_disable	disable software event register write
evic_register_write_enable_get	get software event register write enable status
evic_bit_write_enable	enable software event generation bit write
evic_bit_write_disable	disable software event generation bit write
evic_bit_write_enable_get	get software event generation bit write enable status
evic_software_event_generation	evic software event generation

## Enum event\_source\_enum

**Table 3-333. Enum event\_source\_enum**

Member name	Function description
EVIC_SOURCE_DISABLED	Event signal output is disabled
EVIC_SOURCE_SOFTWARE	Software generation event
EVIC_SOURCE_SGIO_INPUT_DETECTION0	Single I/O input detection 0
EVIC_SOURCE_SGIO_INPUT_DETECTION1	Single I/O input detection 1
EVIC_SOURCE_SGIO_INPUT_DETECTION2	Single I/O input detection 2
EVIC_SOURCE_SGIO_INPUT_DETECTION3	Single I/O input detection 3
EVIC_SOURCE_GRP_E_INPUT_DETECTION	Group E I/O input detection
EVIC_SOURCE_GRP_F_INPUT_DETECTION	Group F I/O input detection
EVIC_SOURCE_TIMER0_OVERFLOW	TIMER0 overflow event
EVIC_SOURCE_TIMER0_UNDERFLOW	TIMER0 underflow event
EVIC_SOURCE_TIMER0_CH0_COMPARE	TIMER0_CH0 compare event
EVIC_SOURCE_TIMER0_CH1_COMPARE	TIMER0_CH1 compare event
EVIC_SOURCE_TIMER0_CH2_COMPARE	TIMER0_CH2 compare event
EVIC_SOURCE_TIMER0_CH3_COMPARE	TIMER0_CH3 compare event
EVIC_SOURCE_TIMER0_MCH0_COMPARE	TIMER0_MCH0 compare event
EVIC_SOURCE_TIMER0_MCH1_COMPARE	TIMER0_MCH1 compare event
EVIC_SOURCE_TIMER0_MCH2_COMPARE	TIMER0_MCH2 compare event
EVIC_SOURCE_TIMER0_MCH3_COMPARE	TIMER0_MCH3 compare event
EVIC_SOURCE_TIMER0_TRGO	TIMER0 TRGO signal
EVIC_SOURCE_TIMER7_OVERFLOW	TIMER7 overflow event
EVIC_SOURCE_TIMER7_UNDERFLOW	TIMER7 underflow event
EVIC_SOURCE_TIMER7_CH0_COMPARE	TIMER7_CH0 compare event

Member name	Function description
EVIC_SOURCE_TIMER7_CH1_COMPARE	TIMER7_CH1 compare event
EVIC_SOURCE_TIMER7_CH2_COMPARE	TIMER7_CH2 compare event
EVIC_SOURCE_TIMER7_CH3_COMPARE	TIMER7_CH3 compare event
EVIC_SOURCE_TIMER7_MCH0_COMPARE	TIMER7_MCH0 compare event
EVIC_SOURCE_TIMER7_MCH1_COMPARE	TIMER7_MCH1 compare event
EVIC_SOURCE_TIMER7_MCH2_COMPARE	TIMER7_MCH2 compare event
EVIC_SOURCE_TIMER7_MCH3_COMPARE	TIMER7_MCH3 compare event
EVIC_SOURCE_TIMER7_TRGO	TIMER7 TRGO signal
EVIC_SOURCE_TIMER1_CH0_COMPARE	TIMER1_CH0 compare event
EVIC_SOURCE_TIMER1_CH1_COMPARE	TIMER1_CH1 compare event
EVIC_SOURCE_TIMER1_CH2_COMPARE	TIMER1_CH2 compare event
EVIC_SOURCE_TIMER1_CH3_COMPARE	TIMER1_CH3 compare event
EVIC_SOURCE_TIMER1_OVERFLOW	TIMER1 overflow event
EVIC_SOURCE_TIMER1_UNDERFLOW	TIMER1 underflow event
EVIC_SOURCE_TIMER1_TRGO	TIMER1 TRGO signal
EVIC_SOURCE_TIMER2_CH0_COMPARE	TIMER2_CH0 compare event
EVIC_SOURCE_TIMER2_CH1_COMPARE	TIMER2_CH1 compare event
EVIC_SOURCE_TIMER2_CH2_COMPARE	TIMER2_CH2 compare event
EVIC_SOURCE_TIMER2_CH3_COMPARE	TIMER2_CH3 compare event
EVIC_SOURCE_TIMER2_OVERFLOW	TIMER2 overflow event
EVIC_SOURCE_TIMER2_UNDERFLOW	TIMER2 underflow event
EVIC_SOURCE_TIMER2_TRGO	TIMER2 TRGO signal
EVIC_SOURCE_CPTIMER0_COUNTER0_OVERFLOW	CPTIMER0 counter 0 overflow event
EVIC_SOURCE_CPTIMERW_OCH0_COMPARE_MATCH	CPTIMERW_OCH0 compare match event
EVIC_SOURCE_CPTIMERW_OCH1_COMPARE_MATCH	CPTIMERW_OCH1 compare match event
EVIC_SOURCE_CPTIMERW_OCH2_COMPARE_MATCH	CPTIMERW_OCH2 compare match event
EVIC_SOURCE_CPTIMERW_OCH3_COMPARE_MATCH	CPTIMERW_OCH3 compare match event
EVIC_SOURCE_GPTIMER0_CH0_COMPARE_MATCH	GPTIMER0_CH0 compare match event
EVIC_SOURCE_GPTIMER0_CH1_COMPARE_MATCH	GPTIMER0_CH1 compare match event
EVIC_SOURCE_GPTIMER0_CH0_ADD_COMPARE_MATCH	GPTIMER0_CH0 addition compare match event
EVIC_SOURCE_GPTIMER0_CH1_ADD_COMPARE_MATCH	GPTIMER0_CH1 addition compare match event
EVIC_SOURCE_GPTIMER0_OVERFLOW	GPTIMER0 overflow event
EVIC_SOURCE_GPTIMER0_UNDERFLOW	GPTIMER0 underflow event

Member name	Function description
EVIC_SOURCE_GPTIMER0_ADTCV1_MATCH	GPTIMER0 ADTCV1 match event
EVIC_SOURCE_GPTIMER0_ADTCV2_MATCH	GPTIMER0 ADTCV2 match event
EVIC_SOURCE_GPTIMER0_REPEAT_COUNT_END	GPTIMER0 end of repeat count event
EVIC_SOURCE_CMP0_OUT	CMP0 output signal
EVIC_SOURCE_CMP1_OUT	CMP1 output signal
EVIC_SOURCE_CMP2_OUT	CMP2 output signal
EVIC_SOURCE_CMP3_OUT	CMP3 output signal
EVIC_SOURCE_ADC0_GRP_SCAN_COMPLETE	ADC0 group scan complete event
EVIC_SOURCE_ADC2_GRP_SCAN_COMPLETE	ADC2 group scan complete event
EVIC_SOURCE_LVD1_VOLTAGE_DETECTION	LVD1 voltage detection
EVIC_SOURCE_LVD2_VOLTAGE_DETECTION	LVD2 voltage detection
EVIC_SOURCE_FWDGT_UNDERFLOW_OR_REFRESH	FWDGT underflow or refresh
EVIC_SOURCE_HXTAL_STUCK	HXTAL stuck event
EVIC_SOURCE_DMA0_CH0_TRANSFER_FINISH	DMA0 channel 0 full transfer finish event
EVIC_SOURCE_DMA0_CH1_TRANSFER_FINISH	DMA0 channel 1 full transfer finish event
EVIC_SOURCE_DMA0_CH2_TRANSFER_FINISH	DMA0 channel 2 full transfer finish event
EVIC_SOURCE_DMA0_CH3_TRANSFER_FINISH	DMA0 channel 3 full transfer finish event
EVIC_SOURCE_DMA0_CH4_TRANSFER_FINISH	DMA0 channel 4 full transfer finish event
EVIC_SOURCE_DMA0_CH5_TRANSFER_FINISH	DMA0 channel 5 full transfer finish event
EVIC_SOURCE_DMA1_CH0_TRANSFER_FINISH	DMA1 channel 0 full transfer finish event
EVIC_SOURCE_DMA1_CH1_TRANSFER_FINISH	DMA1 channel 1 full transfer finish event
EVIC_SOURCE_DMA1_CH2_TRANSFER_FINISH	DMA1 channel 2 full transfer finish event
EVIC_SOURCE_DMA1_CH3_TRANSFER_FINISH	DMA1 channel 3 full transfer finish event
EVIC_SOURCE_DMA1_CH4_TRANSFER_FINISH	DMA1 channel 4 full transfer finish event
EVIC_SOURCE_DMA1_CH5_TRANSFER_FINISH	DMA1 channel 5 full transfer finish event

Member name	Function description
SH	
EVIC_SOURCE_CAN_TRANSMIT_INT	CAN transmit interrupt event
EVIC_SOURCE_CAN_FIFO0_INT	CAN FIFO0 interrupt event
EVIC_SOURCE_CAN_FIFO1_INT	CAN FIFO1 interrupt event
EVIC_SOURCE_CAN_EWMC_INT	CAN EWMC interrupt event
EVIC_SOURCE_UART0_ERROR	UART0 error (PERR /FERR /NERR /ORERR)
EVIC_SOURCE_UART0_TRANSMIT_COMPLETE	UART0 transmission complete
EVIC_SOURCE_UART1_ERROR	UART1 error (PERR /FERR /NERR /ORERR)
EVIC_SOURCE_UART1_TRANSMIT_COMPLETE	UART1 transmission complete
EVIC_SOURCE_UART2_ERROR	UART2 error (PERR /FERR /NERR /ORERR)
EVIC_SOURCE_UART2_TRANSMIT_COMPLETE	UART2 transmission complete
EVIC_SOURCE_UART3_ERROR	UART3 error (PERR /FERR /NERR /ORERR)
EVIC_SOURCE_UART3_TRANSMIT_COMPLETE	UART3 transmission complete
EVIC_SOURCE_SPI_ERROR	SPI error
EVIC_SOURCE_SPI_COMMUNICATION_COMPLETE	SPI transfer complete event
EVIC_SOURCE_I2C_COMMUNICATION_ERROR_OR_EVENT	I2C communication error/communication event
EVIC_SOURCE_I2C_RECEIVE_DATA_NOT_EMPTY	I2C_RDATA is not empty during receiving
EVIC_SOURCE_I2C_TRANSMIT_INT	I2C transmit interrupt event
EVIC_SOURCE_I2C_TRANSFER_COMPLETE	I2C transfer complete event
EVIC_SOURCE_GPTIMER1_CH0_COMPARE_MATCH	GPTIMER1_CH0 compare match event
EVIC_SOURCE_GPTIMER0_UNDERFLOW	GPTIMER1_CH1 compare match event
EVIC_SOURCE_GPTIMER0_ADTCV1_MATCH	GPTIMER1_CH0 addition compare match event
EVIC_SOURCE_GPTIMER0_ADTCV2_MATCH	GPTIMER1_CH1 addition compare match event
EVIC_SOURCE_GPTIMER0_REPEAT_COUNT_END	GPTIMER1 overflow event
EVIC_SOURCE_CMP0_OUT	GPTIMER1 underflow event
EVIC_SOURCE_CMP1_OUT	GPTIMER1 ADTCV1 match event
EVIC_SOURCE_CMP2_OUT	GPTIMER1 ADTCV2 match event
EVIC_SOURCE_CMP3_OUT	GPTIMER1 end of repeat count event

### Enum evic\_periph\_enum

Table 3-334. Enum evic\_periph\_enum

Member name	Function description
EVENT_INTERCONNECT_SGIO0	single I/O 0

Member name	Function description
EVENT_INTERCONNECT_SGIO1	single I/O 1
EVENT_INTERCONNECT_SGIO2	single I/O 2
EVENT_INTERCONNECT_SGIO3	single I/O 3
EVENT_INTERCONNECT_INGRPE	input group E
EVENT_INTERCONNECT_INGRPF	input group F
EVENT_INTERCONNECT_OUTGRPE	output group E
EVENT_INTERCONNECT_OUTGRPF	output group F
EVENT_INTERCONNECT_DAC0_OUT0_COV	DAC0_OUT0 conversion
EVENT_INTERCONNECT_DAC0_OUT1_COV	DAC0_OUT1 conversion
EVENT_INTERCONNECT_ADC0_GRP_EXTRIG0	ADC0 group external trigger 0
EVENT_INTERCONNECT_ADC0_GRP_EXTRIG1	ADC0 group external trigger 1
EVENT_INTERCONNECT_ADC2_GRP_EXTRIG0	ADC2 group external trigger 0
EVENT_INTERCONNECT_ADC2_GRP_EXTRIG1	ADC2 group external trigger 1
EVENT_INTERCONNECT_RCU	clock switch to IRC32M
EVENT_INTERCONNECT_TIMER0	TIMER0
EVENT_INTERCONNECT_TIMER7	TIMER7
EVENT_INTERCONNECT_CPTIMER0	CPTIMER0
EVENT_INTERCONNECT_CPTIMERW	CPTIMERW
EVENT_INTERCONNECT_TIMER1	TIMER1
EVENT_INTERCONNECT_TIMER2	TIMER2
EVENT_INTERCONNECT_GPTIMER_T_RIGIN0	GPTIMER trigger input 0
EVENT_INTERCONNECT_GPTIMER_T_RIGIN1	GPTIMER trigger input 1
EVENT_INTERCONNECT_GPTIMER_T_RIGIN2	GPTIMER trigger input 2
EVENT_INTERCONNECT_GPTIMER_T_RIGIN3	GPTIMER trigger input 3
EVENT_INTERCONNECT_GPTIMER_T_RIGIN4	GPTIMER trigger input 4
EVENT_INTERCONNECT_GPTIMER_T_RIGIN5	GPTIMER trigger input 5
EVENT_INTERCONNECT_GPTIMER_T_RIGIN6	GPTIMER trigger input 6
EVENT_INTERCONNECT_GPTIMER_T_RIGIN7	GPTIMER trigger input 7

Member name	Function description
EVENT_INTERCONNECT_DAC0_OUT0_EN	DAC0_OUT0 enable
EVENT_INTERCONNECT_DAC0_OUT1_EN	DAC0_OUT1 enable
EVENT_INTERCONNECT_DMAMUX_0	DMAMUX multiplexer input 0
EVENT_INTERCONNECT_DMAMUX_1	DMAMUX multiplexer input 1
EVENT_INTERCONNECT_DMAMUX_2	DMAMUX multiplexer input 2
EVENT_INTERCONNECT_DMAMUX_3	DMAMUX multiplexer input 3
EVENT_INTERCONNECT_DMAMUX_4	DMAMUX multiplexer input 4
EVENT_INTERCONNECT_DMAMUX_5	DMAMUX multiplexer input 5
EVENT_INTERCONNECT_DMAMUX_6	DMAMUX multiplexer input 6
EVENT_INTERCONNECT_DMAMUX_7	DMAMUX multiplexer input 7
EVENT_INTERCONNECT_DMAMUX_8	DMAMUX multiplexer input 8
EVENT_INTERCONNECT_DMAMUX_9	DMAMUX multiplexer input 9
EVENT_INTERCONNECT_DMAMUX_10	DMAMUX multiplexer input 10
EVENT_INTERCONNECT_DMAMUX_11	DMAMUX multiplexer input 11

### Enum evic\_cptimer\_enum

Table 3-335. Enum evic\_cptimer\_enum

Member name	Function description
TARGET_CPTIMERW	CPTIMERW
TARGET_CPTIMER0	CPTIMER0

### Enum evic\_single\_io\_enum

Table 3-336. Enum evic\_single\_io\_enum

Member name	Function description
SGIO0	single I/O instance 0
SGIO1	single I/O instance 1
SGIO2	single I/O instance 2
SGIO3	single I/O instance 3

### evic\_init

The description of evic\_init is shown as below:

Table 3-337. Function evic\_init

Function name	evic_init
Function prototype	void evic_init(evic_periph_enum target_periph, event_source_enum event_source);
Function descriptions	set the event source for target peripheral
Precondition	-
The called functions	-

Input parameter{in}	
target_periph	refer to enum <a href="#">Table 3-334. Enum evic_periph_enum</a>
Input parameter{in}	
event_source	refer to enum <a href="#">Table 3-333. Enum event_source_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the software event for TIMER0 */
```

```
evic_init(EVIC_SOURCE_SOFTWARE, EVENT_INTERCONNECT_TIMER0);
```

### evic\_event\_source\_get

The description of evic\_event\_source\_get is shown as below:

**Table 3-338. Function evic\_event\_source\_get**

Function name	evic_event_source_get
Function prototype	event_source_enum evic_event_source_get(evic_periph_enum target_periph);
Function descriptions	get the trigger input signal for target peripheral
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	refer to enum <a href="#">Table 3-334. Enum evic_periph_enum</a>
Output parameter{out}	
-	-
Return value	
event_source	refer to enum <a href="#">Table 3-333. Enum event_source_enum</a>

Example:

```
/* get the trigger input signal for TIMER0 */
```

```
event_source_enum event_source;
```

```
event_source = evic_event_source_get(EVENT_INTERCONNECT_TIMER0);
```

### evic\_register\_lock\_set

The description of evic\_register\_lock\_set is shown as below:

**Table 3-339. Function evic\_register\_lock\_set**

Function name	evic_register_lock_set
Function prototype	void evic_register_lock_set(evic_periph_enum target_periph);
Function descriptions	lock the event interconnect register

Precondition	-
The called functions	-
Input parameter{in}	
target_periph	refer to enum <a href="#">Table 3-334. Enum evic_periph_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* lock the event interconnect for TIMER0 register */
```

```
evic_register_lock_set(EVENT_INTERCONNECT_TIMER0);
```

### evic\_register\_lock\_get

The description of evic\_register\_lock\_get is shown as below:

**Table 3-340. Function evic\_register\_lock\_get**

Function name	evic_register_lock_get
Function prototype	FlagStatus evic_register_lock_get(evic_periph_enum target_periph);
Function descriptions	get the event interconnect register lock status
Precondition	-
The called functions	-
Input parameter{in}	
target_periph	refer to enum <a href="#">Table 3-334. Enum evic_periph_enum</a>
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the event interconnect for TIMER0 register lock status */
```

```
FlagStatus status;
```

```
status = evic_register_lock_get(EVENT_INTERCONNECT_TIMER0);
```

### evic\_cptimer\_slave\_mode\_select

The description of evic\_cptimer\_slave\_mode\_select is shown as below:

**Table 3-341. Function evic\_cptimer\_slave\_mode\_select**

Function name	evic_cptimer_slave_mode_select
Function prototype	void evic_cptimer_slave_mode_select(evic_cptimer_enum cptimer_periph, uint32_t slavemode);
Function descriptions	select CPTIMERW or CPTIMER0 slave mode



<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cptimer_periph</b>	refer to enum <a href="#">Table 3-335. Enum evic_cptimer_enum</a>
<b>Input parameter{in}</b>	
<b>slavemode</b>	slave mode selection
<b>EVIC_COUNTER_ENABLE</b>	counter is enabled
<b>EVIC_COUNTER_RESTART</b>	counter is restarted
<b>EVIC_EVENT_COUNT</b>	event count
<b>EVIC_EVENT_TRIGGER_DISABLE</b>	event trigger is disabled
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select event count as CPTIMERW slave mode */
```

```
evic_cptimer_slave_mode_select(TARGET_CPTIMERW, EVIC_EVENT_COUNT);
```

### evic\_group\_member\_config

The description of evic\_group\_member\_config is shown as below:

**Table 3-342. Function evic\_group\_member\_config**

<b>Function name</b>	evic_group_member_config
<b>Function prototype</b>	void evic_group_member_config(uint32_t group_pin, uint32_t group_port);
<b>Function descriptions</b>	configure evic group member
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>group_pin</b>	group I/O pin
<b>EVIC_GRPIO_PIN_x</b>	pin x (x=8..14)
<b>Input parameter{in}</b>	
<b>group_port</b>	group port selection
<b>EVIC_GRPIO_PORT_E</b>	group I/O port E is selected
<b>EVIC_GRPIO_PORT_F</b>	group I/O port F is selected
<b>Input parameter{in}</b>	
<b>newvalue</b>	ENALBE or DISABLE
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* configure evic group E member */
```

```
evic_group_member_config(EVIC_GRP_GPIO_PIN_8|EVIC_GRP_GPIO_PIN_9,EVIC_GRP_GPIO_PORT_E, ENALBE);
```

### evic\_group\_edge\_detection\_config

The description of evic\_group\_edge\_detection\_config is shown as below:

**Table 3-343. Function evic\_group\_edge\_detection\_config**

<b>Function name</b>	evic_group_edge_detection_config
<b>Function prototype</b>	void evic_group_edge_detection_config(uint32_t group_edge, uint32_t group_port);
<b>Function descriptions</b>	configure evic group input detection edge
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>group_edge</b>	group I/O input detection edge
EVIC_GRP_GPIO_DETECT_ION_RISING	generate event when group I/O input rising edge
EVIC_GRP_GPIO_DETECT_ION_FALLING	generate event when group I/O input falling edge
EVIC_GRP_GPIO_DETECT_ION_BOTH_EDGE	generate event when group I/O input both edges
<b>Input parameter{in}</b>	
<b>group_port</b>	group port selection
EVIC_GRP_GPIO_PORT_E	group I/O port E is selected
EVIC_GRP_GPIO_PORT_F	group I/O port F is selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure evic group E input detection edge */
```

```
evic_group_member_config(EVIC_GRP_GPIO_DETECT_RISING,EVIC_GRP_GPIO_PORT_E);
```

### evic\_group\_output\_level\_config

The description of evic\_group\_output\_level\_config is shown as below:

Table 3-344. Function `evic_group_output_level_config`

<b>Function name</b>	<code>evic_group_output_level_config</code>
<b>Function prototype</b>	<code>void evic_group_output_level_config(uint32_t group_level, uint32_t group_port);</code>
<b>Function descriptions</b>	configure evic group output level
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>group_level</b>	group I/O output level
<code>EVIC_GRPPIO_OUTPU T_LOW</code>	group I/O output low level when event comes
<code>EVIC_GRPPIO_OUTPU T_HIGH</code>	group I/O output high level when event comes
<code>EVIC_GRPPIO_OUTPU T_INVERTED</code>	group I/O output inverted level when event comes
<code>EVIC_GRPPIO_OUTPU T_DATA</code>	group I/O output holding data when event comes
<code>EVIC_GRPPIO_OUTPU T_CIRCULAR_DATA</code>	group I/O circularly shift output holding data when event comes
<b>Input parameter{in}</b>	
<b>group_port</b>	group port selection
<code>EVIC_GRPPIO_PORT_E</code>	group I/O port E is selected
<code>EVIC_GRPPIO_PORT_F</code>	group I/O port F is selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure evic group E output level */
```

```
evic_group_output_level_config(EVIC_GRPPIO_OUTPUT_LOW,EVIC_GRPPIO_PORT_E);
```

### **evic\_group\_overwrite\_enable**

The description of `evic_group_overwrite_enable` is shown as below:

Table 3-345. Function `evic_group_overwrite_enable`

<b>Function name</b>	<code>evic_group_overwrite_enable</code>
<b>Function prototype</b>	<code>void evic_group_overwrite_enable(uint32_t group_port);</code>
<b>Function descriptions</b>	enable evic group overwrite function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>group_port</b>	group port selection

EVIC_GRPIO_PORT_E	group I/O port E is selected
EVIC_GRPIO_PORT_F	group I/O port F is selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable evic group E overwrite function */
```

```
evic_group_overwrite_enable(EVIC_GRPIO_PORT_E);
```

### evic\_group\_overwrite\_disable

The description of evic\_group\_overwrite\_disable is shown as below:

**Table 3-346. Function evic\_group\_overwrite\_disable**

Function name	evic_group_overwrite_disable
Function prototype	void evic_group_overwrite_disable (uint32_t group_port);
Function descriptions	disable evic group overwrite function
Precondition	-
The called functions	-
Input parameter{in}	
group_port	group port selection
EVIC_GRPIO_PORT_E	group I/O port E is selected
EVIC_GRPIO_PORT_F	group I/O port F is selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable evic group E overwrite function */
```

```
evic_group_overwrite_disable(EVIC_GRPIO_PORT_E);
```

### evic\_data\_set

The description of evic\_data\_set is shown as below:

**Table 3-347. Function evic\_data\_set**

Function name	evic_data_set
Function prototype	void evic_data_set(uint8_t data, uint32_t group_port);
Function descriptions	set group holding data value
Precondition	-
The called functions	-

Input parameter{in}	
<b>data</b>	data to be loaded (0~127)
Input parameter{in}	
<b>group_port</b>	group port selection
<i>EVIC_GRPIO_PORT_E</i>	group I/O port E is selected
<i>EVIC_GRPIO_PORT_F</i>	group I/O port F is selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set group E holding data value */
evic_data_set(0xF, EVIC_GRPIO_PORT_E);
```

### evic\_data\_get

The description of evic\_data\_get is shown as below:

**Table 3-348. Function evic\_data\_get**

<b>Function name</b>	evic_data_get
<b>Function prototype</b>	uint8_t evic_data_get(uint32_t group_port);
<b>Function descriptions</b>	get group holding data value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>group_port</b>	group port selection
<i>EVIC_GRPIO_PORT_E</i>	group I/O port E is selected
<i>EVIC_GRPIO_PORT_F</i>	group I/O port F is selected
Output parameter{out}	
-	-
Return value	
<b>holding data</b>	0x00~0x7F

Example:

```
/* get group E holding data value */
uint8_t data;
data = evic_data_get(EVIC_GRPIO_PORT_E);
```

### evic\_single\_io\_config

The description of evic\_single\_io\_config is shown as below:

Table 3-349. Function `evic_single_io_config`

<b>Function name</b>	<code>evic_single_io_config</code>
<b>Function prototype</b>	<code>void evic_single_io_config(uint32_t single_pin, uint32_t single_port, evic_single_io_enum single_io);</code>
<b>Function descriptions</b>	configure evic single I/O
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>single_pin</b>	single I/O pin
<code>EVIC_SGIO_PIN_x</code>	pin x (x=8..14)
<b>Input parameter{in}</b>	
<b>group_port</b>	group port selection
<code>EVIC_SGIO_PORT_E</code>	group I/O port E is selected
<code>EVIC_SGIO_PORT_F</code>	group I/O port F is selected
<b>Input parameter{in}</b>	
<b>single_io</b>	single I/O instance
<code>SGIOx</code>	single I/O instance x (x=0..3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure evic single I/O instance 0 */
```

```
evic_single_io_config(EVIC_SGIO_PIN_1, EVIC_SGIO_PORT_F, SGIO0);
```

### **`evic_single_io_edge_detection_config`**

The description of `evic_single_io_edge_detection_config` is shown as below:

Table 3-350. Function `evic_single_io_config`

<b>Function name</b>	<code>evic_single_io_edge_detection_config</code>
<b>Function prototype</b>	<code>void evic_single_io_edge_detection_config(uint32_t single_edge, evic_single_io_enum single_io);</code>
<b>Function descriptions</b>	configure evic single I/O input detection edge
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>single_edge</b>	single I/O input detection edge
<code>EVIC_SGIO_DETECTI ON_RISING</code>	generate event when single I/O input rising edge
<code>EVIC_SGIO_DETECTI ON_FALLING</code>	generate event when single I/O input falling edge
<code>EVIC_SGIO_DETECTI</code>	generate event when single I/O input both edges

<i>ON_BOTH_EDGE</i>	
<b>Input parameter{in}</b>	
<b>single_io</b>	single I/O instance
<i>SGIOx</i>	single I/O instance x (x=0..3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure evic single I/O instance 0 input detection edge */
```

```
evic_single_io_edge_detection_config(EVIC_SGIO_DETECTION_RISING, SGIO0);
```

### evic\_single\_io\_output\_level\_config

The description of `evic_single_io_output_level_config` is shown as below:

**Table 3-351. Function `evic_single_io_output_level_config`**

<b>Function name</b>	<code>evic_single_io_output_level_config</code>
<b>Function prototype</b>	<code>void evic_single_io_output_level_config(uint32_t single_level, evic_single_io_enum single_io);</code>
<b>Function descriptions</b>	configure evic single I/O output level
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>single_level</b>	single I/O output level
<i>EVIC_SGIO_OUTPUT_LOW</i>	single I/O output low level when event comes
<i>EVIC_SGIO_OUTPUT_HIGH</i>	single I/O output high level when event comes
<i>EVIC_SGIO_OUTPUT_INVERTED</i>	single I/O output inverted level when event comes
<b>Input parameter{in}</b>	
<b>single_io</b>	single I/O instance
<i>SGIOx</i>	single I/O instance x (x=0..3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure evic single I/O instance 0 output level */
```

```
evic_single_io_output_level_config(EVIC_SGIO_OUTPUT_LOW, SGIO0);
```

## evic\_register\_write\_enable

The description of evic\_register\_write\_enable is shown as below:

**Table 3-352. Function evic\_register\_write\_enable**

<b>Function name</b>	evic_register_write_enable
<b>Function prototype</b>	void evic_register_write_enable(void);
<b>Function descriptions</b>	enable software event register 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 software event register write */
void evic_register_write_enable(void);
```

## evic\_register\_write\_disable

The description of evic\_register\_write\_disable is shown as below:

**Table 3-353. Function evic\_register\_write\_disable**

<b>Function name</b>	evic_register_write_disable
<b>Function prototype</b>	void evic_register_write_disable(void);
<b>Function descriptions</b>	disable software event register 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 software event register write */
void evic_register_write_disable(void);
```

## evic\_register\_write\_enable\_get

The description of evic\_register\_write\_enable\_get is shown as below:



Table 3-354. Function `evic_register_write_enable_get`

Function name	<code>evic_register_write_enable_get</code>
Function prototype	<code>FlagStatus evic_register_write_enable_get(void);</code>
Function descriptions	get software event register write enable status
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get software event register write enable status */
```

```
FlagStatus status;
```

```
status = evic_register_write_enable_get(void);
```

### **evic\_bit\_write\_enable**

The description of `evic_bit_write_enable` is shown as below:

Table 3-355. Function `evic_bit_write_enable`

Function name	<code>evic_bit_write_enable</code>
Function prototype	<code>void evic_bit_write_enable(void);</code>
Function descriptions	enable software event generation bit write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable software event generation bit write */
```

```
void evic_bit_write_enable(void);
```

### **evic\_bit\_write\_disable**

The description of `evic_bit_write_disable` is shown as below:

Table 3-356. Function `evic_bit_write_disable`

Function name	<code>evic_bit_write_disable</code>
Function prototype	<code>void evic_bit_write_disable(void);</code>
Function descriptions	disable software event generation bit write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable software event generation bit write */
void evic_bit_write_disable(void);
```

### **`evic_bit_write_enable_get`**

The description of `evic_bit_write_enable_get` is shown as below:

Table 3-357. Function `evic_register_write_enable_get`

Function name	<code>evic_bit_write_enable_get</code>
Function prototype	<code>FlagStatus evic_bit_write_enable_get(void);</code>
Function descriptions	get software event generation bit write enable status
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get software event generation bit write enable status */
FlagStatus status;
status = evic_bit_write_enable_get(void);
```

### **`evic_software_event_generation`**

The description of `evic_software_event_generation` is shown as below:

**Table 3-358. Function evic\_software\_event\_generation**

<b>Function name</b>	evic_software_event_generation
<b>Function prototype</b>	void evic_software_event_generation(void);
<b>Function descriptions</b>	evic software event generation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* evic software event generation */

void evic_software_event_generation(void);
```

## 3.13. EXTI

EXTI is the interrupt/event controller in the MCU. It contains up to 25 independent edge detectors and generates interrupt requests or events to the processor. The EXTI registers are listed in chapter [3.13.1](#), 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 3-359. EXTI Registers**

Registers	Descriptions
EXTI_INTEN	interrupt enable register
EXTI_EVEN	event enable register
EXTI_RTEN	rising edge trigger enable register
EXTI_FTEN	falling edge trigger enable register
EXTI_SWIEV	software interrupt event register
EXTI_PD	pending register
EXTI_DFEN	digital filter enable register
EXTI_SCS	sampling clock select register

### 3.13.2. Descriptions of Peripheral functions

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

Table 3-360. 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 EXTI software interrupt event from EXTI line x
exti_software_interrupt_disable	disable EXTI software interrupt event from EXTI line x
exti_digital_filter_enable	enable the digital filter from EXTI line x
exti_digital_filter_disable	disable the digital filter 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 3-361. 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

enum name	Function description
EXTI_22	EXTI line 22
EXTI_23	EXTI line 23
EXTI_24	EXTI line 24

### Enum exti\_mode\_enum

Table 3-362. exti\_mode\_enum

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

### Enum exti\_trig\_type\_enum

Table 3-363. 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 3-364. 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 3-365. 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 3-361. exti_line_enum</a>
Input parameter{in}	
mode	EXTI mode, refer to <a href="#">Table 3-362. exti_mode_enum</a>
Input parameter{in}	
trig_type	trigger type, refer to <a href="#">Table 3-363. 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 3-366. 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 3-361. 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 3-367. 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 3-361. 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 3-368. 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 3-361. 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 3-369. Function exti\_event\_disable

Function name	exti_event_disable
Function prototype	void exti_event_disable(exti_line_enum linex);
Function descriptions	disable the events from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-361. exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### exti\_software\_interrupt\_enable

The description of exti\_software\_interrupt\_enable is shown as below:

Table 3-370. Function exti\_software\_interrupt\_enable

Function name	exti_software_interrupt_enable
Function prototype	void exti_software_interrupt_enable(exti_line_enum linex);
Function descriptions	enable software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-361. exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### exti\_software\_interrupt\_disable

The description of exti\_software\_interrupt\_disable is shown as below:

Table 3-371. Function exti\_software\_interrupt\_disable

Function name	exti_software_interrupt_disable
---------------	---------------------------------



<b>Function prototype</b>	void exti_software_interrupt_disable(exti_line_enum linex);
<b>Function descriptions</b>	disable 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 3-361. 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\_digital\_filter\_enable

The description of exti\_digital\_filter\_enable is shown as below:

**Table 3-372. Function exti\_digital\_filter\_enable**

<b>Function name</b>	exti_digital_filter_enable
<b>Function prototype</b>	void exti_digital_filter_enable(exti_line_enum linex, uint32_t pclk_div);
<b>Function descriptions</b>	enable the digital filter 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 3-361. exti_line_enum</a>
<b>Input parameter{in}</b>	
<b>pclk_div</b>	sampling clock select
EXTI_SAMPLING_PCLK_DIV1	sampling clock select PCLK
EXTI_SAMPLING_PCLK_DIV8	sampling clock select PCLK / 8
EXTI_SAMPLING_PCLK_DIV32	sampling clock select PCLK / 32
EXTI_SAMPLING_PCLK_DIV64	sampling clock select PCLK / 64
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable EXTI line 0 digital filter */
```

```
exti_digital_filter_enable(EXTI_0, EXTI_SAMPLING_PCLK_DIV1);
```

### exti\_digital\_filter\_disable

The description of exti\_digital\_filter\_disable is shown as below:

**Table 3-373. Function exti\_digital\_filter\_disable**

<b>Function name</b>	exti_digital_filter_disable
<b>Function prototype</b>	void exti_digital_filter_disable(exti_line_enum linex);
<b>Function descriptions</b>	disable the digital filter 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 3-361. exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable EXTI line 0 digital filter */
exti_digital_filter_disable(EXTI_0);
```

### exti\_flag\_get

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

**Table 3-374. 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 3-361. 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 3-375. 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 3-361. 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 3-376. 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 3-361. 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 3-377. Function exti\_interrupt\_flag\_clear

Function name	exti_interrupt_flag_clear
Function prototype	void exti_interrupt_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-361. exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

## 3.14. FMC

Flash Memory Controller (FMC), which provides all the functionality needed for on-chip flash memory. The FMC registers are listed in chapter [3.14.1](#) the FMC firmware functions are introduced in chapter [3.14.2](#).

### 3.14.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

Table 3-378. FMC Registers

Registers	Descriptions
FMC_WS	FMC wait state 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
FMC_ADDR	FMC address register
FMC_DATA	FMC program data register
FMC_ECCCS	FMC ECC control and status register
FMC_OBSTAT	FMC option bytes status register
FMC_WP	FMC erase/program protection register
FMC_PID0	FMC product ID register 0
FMC_PID1	FMC product ID register 1

### 3.14.2. Descriptions of Peripheral functions

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

**Table 3-379. FMC firmware function**

Function name	Function description
fmc_unlock	unlock the main flash and data flash operation
fmc_lock	lock the main flash and data flash operation
fmc_wscont_set	set the wait state counter value
fmc_prefetch_enable	enable pre-fetch
fmc_prefetch_disable	disable pre-fetch
fmc_ibus_enable	enable IBUS cache
fmc_ibus_disable	disable IBUS cache
fmc_dbus_enable	enable DBUS cache
fmc_dbus_disable	disable DBUS cache
fmc_ibus_reset	reset IBUS cache
fmc_dbus_reset	reset DBUS cache
fmc_blank_check	check whether flash page is blank or not by check blank command
fmc_page_erase	erase main flash or data flash page
fmc_mflash_mass_erase	erase main flash
fmc_dflash_mass_erase	erase data flash
fmc_mass_erase	erase whole chip
fmc_fourword_program	FMC program four words at the corresponding address
fmc_fast_program	FMC fast program one row data starting at the corresponding address
fmc_otp_fourword_program	FMC program a word at the corresponding address in OTP
ob_unlock	unlock the option byte operation
ob_lock	lock the option byte operation
ob_reset	reload the option byte and generate a system reset
ob_erase	erase the FMC option bytes
ob_write_protection_enable	enable write protection
ob_security_protection_config	configure security protection
ob_user_write	program option bytes USER
ob_data_program	program option bytes data
ob_user1_write	program option bytes USER1
ob_wwdgt0_write	program option bytes WWDGT0
ob_wwdgt1_wwdgt2_write	program option bytes WWDGT1 and WWDGT2
ob_fwdgt_write	program option bytes FWDGT
ob_user_get	get OB_USER in register FMC_OBSTAT
ob_data_get	get OB_DATA in register FMC_OBSTAT
ob_write_protection_get	get the FMC option byte write protection (OB_WP) in register FMC_WP

Function name	Function description
ob_security_protection_flag_get	get the FMC option bytes security protection state
fmc_ecc_error_address_get	get the FMC ECC error address in register FMC_ECCCS
fmc_flag_get	get FMC flag status
fmc_flag_clear	clear FMC flag status
fmc_interrupt_enable	enable FMC interrupt
fmc_interrupt_disable	disable FMC interrupt
fmc_interrupt_flag_get	get FMC interrupt flag status
fmc_interrupt_flag_clear	clear FMC interrupt flag status

### Structure optionbyte\_user\_struct

**Table 3-380. Structure optionbyte\_user\_struct**

Member name	Function description
nfdwg_hw	nFWDG_HW option byte USER (OB_FWDGT_HW, OB_FWDGT_SW)
nrst_dpslp	nRST_DPSLP of option byte USER (OB_DEEPSLEEP_RST, OB_DEEPSLEEP_N_RST)
nrst_stdbby	nRST_STDBY of option byte USER (OB_STDBY_RST, OB_STDBY_N_RST)
lvd0en	LVD0EN of option byte USER (OB_LVDOEN_ENABLE, OB_LVDOEN_DISABLE)
fdwgspd_dpslp	FWDGSPD_DPSLP of option byte USER (OB_FWDGDPSLP_ENABLE, OB_FWDGDPSLP_DISABLE)
fdwgspd_stdbby	FWDGSPD_STDBY of option byte USER (OB_FWDGSTANDBY_ENABLE, OB_FWDGSTANDBY_DISABLE)
lvd0t	LVD0T of option byte USER (OB_LVD0T_VALUE0, OB_LVD0T_VALUE1, OB_LVD0T_VALUE2)

### Structure optionbyte\_wwdgt0\_struct

**Table 3-381. Structure optionbyte\_wwdgt0\_struct**

Member name	Function description
psc	psc of option byte WWDGT0 (OB_WWDG_PSC8, OB_WWDG_PSC4, OB_WWDG_PSC2, OB_WWDG_PSC1)
wsp	wsp of option byte WWDGT0 (OB_WSPS_VALUE25, OB_WSPS_VALUE50, OB_WSPS_VALUE75, OB_WSPS_VALUE100)
weps	weps of option byte WWDGT0 (OB_WEPS_VALUE75, OB_WEPS_VALUE50, OB_WEPS_VALUE25, OB_WEPS_VALUE0)
ewie	ewie of option byte WWDGT0 (OB_EWIE_ENABLE, OB_EWIE_DISABLE)

Member name	Function description
wdga	wdga of option byte WWDGT0 (OB_WWDGT_RST_DISABLE, OB_WWDGT_RST_ENABLE)

### Enum fmc\_state\_enum

Table 3-382. fmc\_state\_enum

Enum name	Enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress
FMC_PGSERR	program sequence error
FMC_PGERR	program error
FMC_PGAERR	program alignment error
FMC_WPERR	erase/program protection error
FMC_TOERR	timeout error
FMC_CBCMDERR	the checked area not blank error
FMC_OB_HSPC	high security protection

### Enum fmc\_flag\_enum

Table 3-383. fmc\_flag\_enum

Enum name	Enum description
FMC_FLAG_BUSY	flash busy flag
FMC_FLAG_PGSE RR	program sequence error flag
FMC_FLAG_PGER R	flash program error flag
FMC_FLAG_PGAE RR	flash program alignment error flag
FMC_FLAG_WPER R	flash erase/program protection error flag
FMC_FLAG_END	flash end of operation flag
FMC_FLAG_CBCM DERR	flash checked area by the check blank command is all 0xFF or not flag
FMC_FLAG_ECCC OR	single bit ECC error flag
FMC_FLAG_ECCD ET	double bit ECC error flag
FMC_FLAG_SYSE CC	system memory flash read ECC error flag
FMC_FLAG_MFEC C	main flash read ECC error flag
FMC_FLAG_OTPE CC	OTP read ECC error flag

Enum name	Enum description
FMC_FLAG_OBEC C	OPT read ECC error flag
FMC_FLAG_DFEC C	data flash read ECC error status
FMC_FLAG_SYSE CCBK	ECC error in the first 3K or later 3K region of the information block
FMC_FLAG_OBER R	option bytes read error flag

### Enum fmc\_interrupt\_flag\_enum

Table 3-384. fmc\_interrupt\_flag\_enum

Enum name	Enum description
FMC_INT_FLAG_P GSERR	flash program sequence error interrupt flag
FMC_INT_FLAG_P GERR	flash program error interrupt flag
FMC_INT_FLAG_P GAERR	flash program alignment error interrupt flag
FMC_INT_FLAG_W PERR	flash erase/program protection error interrupt flag
FMC_INT_FLAG_E ND	flash end of operation interrupt flag
FMC_INT_FLAG_C BCMDERR	flash checked area by the check blank command is all 0xFF or not interrupt flag
FMC_INT_FLAG_E CCCOR	single bit ECC error interrupt flag
FMC_INT_FLAG_E CCDET	double bit ECC error interrupt flag

### Enum fmc\_interrupt\_enum

Table 3-385. fmc\_interrupt\_enum

Enum name	Enum description
FMC_INT_ERR	FMC flash error interrupt
FMC_INT_END	FMC flash end of operation interrupt
FMC_INT_ECCDET	FMC double ECC error interrupt
FMC_INT_ECCCO R	FMC single ECC error interrupt



## Enum `fmc_ob_wp_enum`

**Table 3-386. `fmc_ob_wp_enum`**

Enum name	Enum description
OB_WP_NONE	disable all erase/program protection
OB_WP_0	erase/program protection of main flash page 0 ~ 7
OB_WP_1	erase/program protection of main flash page 8 ~ 15
OB_WP_2	erase/program protection of main flash page 16 ~23
OB_WP_3	erase/program protection of main flash page 24 ~ 31
OB_WP_4	erase/program protection of main flash page 32 ~ 39
OB_WP_5	erase/program protection of main flash page 40 ~ 47
OB_WP_6	erase/program protection of main flash page 48 ~ 55
OB_WP_7	erase/program protection of main flash page 56 ~ 63
OB_WP_8	erase/program protection of main flash page 64 ~ 71
OB_WP_9	erase/program protection of main flash page 72 ~ 79
OB_WP_10	erase/program protection of main flash page 80 ~ 87
OB_WP_11	erase/program protection of main flash page 88 ~ 95
OB_WP_12	erase/program protection of main flash page 96 ~ 103
OB_WP_13	erase/program protection of main flash page 104 ~ 111
OB_WP_14	erase/program protection of main flash page 112 ~ 119
OB_WP_15	erase/program protection of main flash page 120 ~ 127
OB_WP_16	erase/program protection of main flash page 128 ~ 135
OB_WP_17	erase/program protection of main flash page 136 ~ 143
OB_WP_18	erase/program protection of main flash page 144 ~ 151
OB_WP_19	erase/program protection of main flash page 152 ~ 159
OB_WP_20	erase/program protection of main flash page 160 ~ 167
OB_WP_21	erase/program protection of main flash page 168 ~ 175
OB_WP_22	erase/program protection of main flash page 176 ~ 183
OB_WP_23	erase/program protection of main flash page 184 ~ 256
OB_WP_24	erase/program protection of data flash page 0 ~ 7
OB_WP_25	erase/program protection of data flash page 8 ~ 15
OB_WP_26	erase/program protection of data flash page 16 ~23
OB_WP_27	erase/program protection of data flash page 24 ~ 31
OB_WP_28	erase/program protection of data flash page 32 ~ 39
OB_WP_29	erase/program protection of data flash page 40 ~ 47
OB_WP_30	erase/program protection of data flash page 48 ~ 55
OB_WP_31	erase/program protection of data flash page 56 ~ 63
OB_WP_ALL	erase/program protection of all pages

## `fmc_unlock`

The description of `fmc_unlock` is shown as below:

Table 3-387. Function fmc\_unlock

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

Example:

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

### fmc\_lock

The description of fmc\_lock is shown as below:

Table 3-388. Function fmc\_lock

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

Example:

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

### fmc\_wscent\_set

The description of fmc\_wscent\_set is shown as below:

Table 3-389. Function fmc\_wscent\_set

Function name	fmc_wscent_set
---------------	----------------

<b>Function prototype</b>	void fmc_wscnt_set(uint32_t wscnt);
<b>Function descriptions</b>	set the wait state counter value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wscnt</b>	wait state counter value
<i>FMC_WAIT_STATE_0</i>	0 wait state added
<i>FMC_WAIT_STATE_1</i>	1 wait state added
<i>FMC_WAIT_STATE_2</i>	2 wait state added
<i>FMC_WAIT_STATE_3</i>	3 wait state added
<i>FMC_WAIT_STATE_4</i>	4 wait state added
<i>FMC_WAIT_STATE_5</i>	5 wait state added
<i>FMC_WAIT_STATE_6</i>	6 wait state added
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set 1wait state */
```

```
fmc_wscnt_set(FMC_WAIT_STATE_1);
```

### fmc\_prefetch\_enable

The description of fmc\_prefetch\_enable is shown as below:

**Table 3-390. 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 3-391. 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\_ibus\_enable

The description of fmc\_ibus\_enable is shown as below:

**Table 3-392. Function fmc\_ibus\_enable**

<b>Function name</b>	fmc_ibus_enable
<b>Function prototype</b>	void fmc_ibus_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 IBUS cache */
fmc_ibus_enable();
```

## fmc\_ibus\_disable

The description of fmc\_ibus\_disable is shown as below:

Table 3-393. Function fmc\_ibus\_disable

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

Example:

```
/* disable IBUS cache */
fmc_ibus_disable();
```

### fmc\_ibus\_reset\_enable

The description of fmc\_ibus\_reset\_enable is shown as below:

Table 3-394. Function fmc\_ibus\_reset\_enable

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

Example:

```
/* enable reset IBUS cache */
fmc_ibus_reset_enable();
```

### fmc\_ibus\_reset\_disable

The description of fmc\_ibus\_reset\_disable is shown as below:

Table 3-395. Function fmc\_ibus\_reset\_disable

Function name	fmc_ibus_reset_disable
---------------	------------------------

<b>Function prototype</b>	void fmc_ibus_reset_disable(void);
<b>Function descriptions</b>	Disable 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:

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

```
fmc_ibus_reset_disable();
```

### fmc\_dbus\_enable

The description of fmc\_dbus\_enable is shown as below:

**Table 3-396. Function fmc\_dbus\_enable**

<b>Function name</b>	fmc_dbus_enable
<b>Function prototype</b>	void fmc_dbus_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 DBUS cache */
```

```
fmc_dbus_enable();
```

### fmc\_dbus\_disable

The description of fmc\_dbus\_disable is shown as below:

**Table 3-397. Function fmc\_dbus\_disable**

<b>Function name</b>	fmc_dbus_disable
<b>Function prototype</b>	void fmc_dbus_disable(void);
<b>Function descriptions</b>	disable DBUS cache

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

Example:

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

```
fmc_dbus_disable();
```

### fmc\_dbus\_reset\_enable

The description of fmc\_dbus\_reset\_enable is shown as below:

**Table 3-398. Function fmc\_dbus\_reset\_enable**

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

Example:

```
/* enable reset DBUS cache */
```

```
fmc_dbus_reset_enable();
```

### fmc\_dbus\_reset\_disable

The description of fmc\_dbus\_reset\_disable is shown as below:

**Table 3-399. Function fmc\_dbus\_reset\_disable**

Function name	fmc_dbus_reset_disable
Function prototype	void fmc_dbus_reset_disable(void);
Function descriptions	disable reset DBUS cache
Precondition	-
The called functions	-

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

Example:

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

```
fmc_dbus_reset_disable();
```

### fmc\_blank\_check

The description of fmc\_blank\_check is shown as below:

**Table 3-400. Function fmc\_blank\_check**

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

Example:

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

```
fmc_state_enum state;
```

```
state = fmc_blank_check(0x8001000, 4);
```



## fmc\_page\_erase

The description of fmc\_page\_erase is shown as below:

**Table 3-401. Function fmc\_page\_erase**

<b>Function name</b>	fmc_page_erase
<b>Function prototype</b>	fmc_state_enum fmc_page_erase(uint32_t page_address);
<b>Function descriptions</b>	FMC erase page
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>page_address</b>	the page address to be erased
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error

Example:

```
/* erase page */

fmc_unlock();

fmc_state_enum state = fmc_page_erase(0x08001000);
```

## fmc\_mflash\_mass\_erase

The description of fmc\_mflash\_mass\_erase is shown as below:

**Table 3-402. Function fmc\_mflash\_mass\_erase**

<b>Function name</b>	fmc_mflash_mass_erase
<b>Function prototype</b>	fmc_state_enum fmc_mflash_mass_erase(void)
<b>Function descriptions</b>	erase main flash
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-

Return value	
<b>fmc_state_enum</b>	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error

Example:

```
/* erase main flash */
```

```
fmc_unlock();
```

```
fmc_state_enum state = fmc_mflash_mass_erase();
```

### **fmc\_dflash\_mass\_erase**

The description of fmc\_dflash\_mass\_erase is shown as below:

**Table 3-403. Function fmc\_dflash\_mass\_erase**

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

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

```
fmc_unlock();
```

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

## fmc\_mass\_erase

The description of fmc\_mass\_erase is shown as below:

**Table 3-404. 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>	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error

Example:

```
/* erase whole chip */
fmc_unlock();

fmc_state_enum state = fmc_mass_erase();
```

## fmc\_fourword\_program

The description of fmc\_fourword\_program is shown as below:

**Table 3-405. Function fmc\_fourword\_program**

<b>Function name</b>	fmc_fourword_program
<b>Function prototype</b>	fmc_state_enum fmc_fourword_program(uint32_t address, uint64_t data_l, uint64_t data_h)
<b>Function descriptions</b>	FMC program four words at the corresponding address
<b>Precondition</b>	fmc_unlock, fmc_page_erase / fmc_mflash_mass_erase / fmc_dflash_mass_erase / fmc_mass_erase
<b>The called functions</b>	-

Input parameter{in}	
<b>address</b>	the address to program
Input parameter{in}	
<b>data_l</b>	low 64-bit to program
Input parameter{in}	
<b>data_h</b>	high 64-bit to program
Output parameter{out}	
-	-
Return value	
<b>fmc_state_enum</b>	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error

Example:

```
/* program 128-bit word at the corresponding address */
```

```
fmc_unlock();
```

```
fmc_page_erase(0x08001000);
```

```
fmc_state_enum state = fmc_fourword_program(0x08001000, 0x0123456789abcdef,
0x08192a3b4c5d6e7f);
```

### fmc\_fast\_program

The description of fmc\_fast\_program is shown as below:

**Table 3-406. Function fmc\_fast\_program**

<b>Function name</b>	fmc_fast_program
<b>Function prototype</b>	fmc_state_enum fmc_fast_program(uint32_t address, uint64_t data[]);
<b>Function descriptions</b>	FMC fast program one row data starting at the corresponding address
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
Input parameter{in}	
<b>address</b>	the address to be programmed
Input parameter{in}	
<b>data</b>	the data to be programmed
Output parameter{out}	
-	-
Return value	

<b>fmc_state_enum</b>	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error

Example:

```
/* data buffer for fast programming */
```

```
static uint64_t data_buffer[32] = {
```

```
    0x0000000000000000U, 0x1111111111111111U, 0x2222222222222222U, 0x3333333333333333U,
```

```
    0x4444444444444444U, 0x5555555555555555U, 0x6666666666666666U, 0x7777777777777777U,
```

```
    0x8888888888888888U, 0x9999999999999999U, 0xAAAAAAAAAAAAAAAAAAU, 0xBBBBBBBBBBBBBBBU,
```

```
    0xCCCCCCCCCCCCCCCCCU, 0xDDDDDDDDDDDDDDDDDU, 0xEEEEEEEEEEEEEEEEU, 0xFFFFFFFFFFFUFFFU,
```

```
    0x0011001100110011U, 0x2233223322332233U, 0x4455445544554455U, 0x6677667766776677U,
```

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

```
    0x2200220022002200U, 0x3311331133113311U, 0x6644664466446644U, 0x7755775577557755U,
```

```
    0xAA88AA88AA88AA88U, 0xBB99BB99BB99BB99U, 0xEECCEECCEECCEECCECU, 0xFFDDFFDDFFDDFFDDU
```

```
};
```

```
fmc_unlock();
```

```
fmc_page_erase(0x08001000);
```

```
/* program flash */
```

```
fmc_state_enum fmc_state = fmc_fast_program(0x08001000, data_buffer);
```

## fmc\_otp\_fourword\_program

The description of fmc\_otp\_fourword\_program is shown as below:

**Table 3-407. Function fmc\_otp\_fourword\_program**

<b>Function name</b>	fmc_otp_fourword_program
<b>Function prototype</b>	fmc_state_enum fmc_otp_fourword_program(uint32_t address, uint64_t data_l, uint64_t data_h)
<b>Function descriptions</b>	FMC program four words at the corresponding address in otp
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>address</b>	the address to program
<b>Input parameter{in}</b>	
<b>data_l</b>	low 64-bit to program
<b>Input parameter{in}</b>	
<b>data_h</b>	high 64-bit to program
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC
<i>FMC_READY</i>	the operation has been completed
<i>FMC_BUSY</i>	the operation is in progress
<i>FMC_PGSERR</i>	program sequence error
<i>FMC_PGERR</i>	program error
<i>FMC_PGAERR</i>	program alignment error
<i>FMC_WPERR</i>	erase/program protection error
<i>FMC_TOERR</i>	timeout error
<i>FMC_CBCMDERR</i>	the checked area not blank error

Example:

```
/* program 128-bit word at the corresponding address */
```

```
fmc_unlock();
```

```
fmc_page_erase(0x08001000);
```

```
fmc_state_enum state = fmc_otp_fourword_program(0x1FFF7000, 0x0123456789abcdef,
0x08192a3b4c5d6e7f);
```

## ob\_unlock

The description of ob\_unlock is shown as below:

**Table 3-408. 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:

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

```
fmc_unlock();
```

```
ob_unlock();
```

### ob\_lock

The description of ob\_lock is shown as below:

**Table 3-409. Function ob\_lock**

<b>Function name</b>	ob_lock
<b>Function prototype</b>	void ob_lock(void);
<b>Function descriptions</b>	lock the option byts 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:

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

```
fmc_unlock();
```

```
ob_lock();
```

### ob\_erase

The description of ob\_erase is shown as below:

Table 3-410. Function ob\_erase

Function name	ob_erase
Function prototype	fmc_state_enum ob_erase(void);
Function descriptions	erase the FMC option bytes
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

Example:

```

/* erase the FMC option bytes */

fmc_state_enum state;

fmc_unlock();

ob_unlock();

state = ob_erase();

```

### ob\_write\_protection\_enable

The description of ob\_write\_protection\_enable is shown as below:

Table 3-411. Function ob\_write\_protection\_enable

Function name	ob_write_protection_enable
Function prototype	fmc_state_enum ob_write_protection_enable(fmc_ob_wp_enum ob_wp);
Function descriptions	enable write protection
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
ob_wp	specify sector to be write protected, refer to <a href="#">Table 3-386. fmc_ob_wp_enum</a>
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

Example:

```

/* enable write protection */

fmc_state_enum state;

```



```
fmc_unlock();

ob_unlock();

state = ob_write_protection_enable(OB_WP_7);
```

### ob\_security\_protection\_config

The description of ob\_security\_protection\_config is shown as below:

**Table 3-412. Function ob\_security\_protection\_config**

<b>Function name</b>	ob_security_protection_config
<b>Function prototype</b>	fmc_state_enum ob_security_protection_config (uint8_t ob_spc);
<b>Function descriptions</b>	configure security protection
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_spc</b>	specify security protection
<i>FMC_NSPC</i>	no security protection
<i>FMC_LSPC</i>	low security protection
<i>FMC_HSPC</i>	high security protection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

Example:

```
/* enable low security protection */

fmc_state_enum state;

ob_unlock();

state = ob_security_protection_config(FMC_LSPC);
```

### ob\_user\_write

The description of ob\_user\_write is shown as below:

**Table 3-413. Function ob\_user\_write**

<b>Function name</b>	ob_user_write
<b>Function prototype</b>	fmc_state_enum ob_user_write(optionbyte_user_struct *optionbyte_user)
<b>Function descriptions</b>	program option bytes USER
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>optionbyte_user</b>	option bytes user structure, the structure members can refer to members of

	the structure refer to <a href="#">Table 3-380. Structure optionbyte_user_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

Example:

```

/* configure user option byte */

optionbyte_user_struct ob_user;

fmc_state_enum state;

fmc_unlock();

ob_unlock();

ob_user.nfwdg_hw = OB_FWDGT_HW;

ob_user.nrst_dpslp = OB_DEEPSLEEP_RST;

ob_user.nrst_stdby = OB_STDBY_RST;

ob_user.lvd0en = OB_LVDOEN_ENABLE;

ob_user.fwdgspd_dpslp = OB_FWDGDPSLP_ENABLE;

ob_user.lvd0t = OB_LVD0T_VALUE0;

state = ob_user_write(ob_user);

```

### ob\_data\_program

The description of ob\_data\_program is shown as below:

**Table 3-414. Function ob\_data\_program**

<b>Function name</b>	ob_data_program
<b>Function prototype</b>	fmc_state_enum ob_data_program(uint16_t ob_data);
<b>Function descriptions</b>	program option bytes DATA
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data</b>	the byte to be programmed
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

Example:

```

/* program option bytes data */

```

```
fmc_state enum state;

fmc_unlock();

ob_unlock();

state = ob_data_program(0x56);
```

### ob\_user1\_write

The description of ob\_user1\_write is shown as below:

**Table 3-415. Function ob\_user1\_write**

Function name	ob_user1_write
Function prototype	fmc_state_enum ob_user1_write(uint8_t ob_wwdg_hw, uint8_t ob_sram_ecc, uint8_t ob_swd_mode, uint8_t ob_nrst_mod)
Function descriptions	program option bytes USER1
Precondition	-
The called functions	-
Input parameter{in}	
<b>ob_wwdg_hw</b>	window watchdog select hardware or software
<i>OB_WWDGT_HW</i>	select hardware window watchdog
<i>OB_WWDGT_SW</i>	select software window watchdog
Input parameter{in}	
<b>ob_sram_parity_chk</b>	sram parity error check
<i>OB_SRAM_ECC_DISAB LE</i>	sram ecc disable
<i>OB_SRAM_ECC_ENAB LE</i>	sram ecc enable
Input parameter{in}	
<b>ob_swd_mode</b>	swd off or on
<i>OB_SWD_DISABLE</i>	swd disable
<i>OB_SWD_ENABLE</i>	swd enable
Input parameter{in}	
<b>ob_nrst_mod</b>	reset input/output
<i>OB_NRST_MODE_VAL UE1</i>	nrst mode = 1
<i>OB_NRST_MODE_VAL UE2</i>	nrst mode = 2
<i>OB_NRST_MODE_VAL UE3</i>	nrst mode = 3
Output parameter{out}	
-	-
Return value	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

```
/* program option bytes USER1 */
```

```
fmc_state enum state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
state = ob_user1_write(OB_WWDGT_HW, OB_SRAM_ECC_DISABLE, OB_SWD_DISABLE,
OB_NRST_MODE_VALUE1);
```

### ob\_wwdgt0\_write

The description of ob\_wwdgt0\_write is shown as below:

**Table 3-416. Function ob\_wwdgt0\_write**

Function name	ob_wwdgt0_write
Function prototype	fmc_state_enum ob_wwdgt0_write(optionbyte_wwdgt0_struct *optionbyte_wwdgt0)
Function descriptions	program option bytes WWDGT0
Precondition	-
The called functions	-
Input parameter{in}	
optionbyte_wwdgt0	option bytes wwdgt0 structure, the structure members can refer to members of the structure refer to <a href="#">Table 3-381. Structure optionbyte_wwdgt0_struct</a>
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

Example:

```
/* program option bytes WWDGT0*/
```

```
optionbyte_wwdgt0_struct ob_wwdgt0;
```

```
fmc_state enum state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
ob_wwdgt0.psc = OB_WWDG_PSC8;
```

```
ob_wwdgt0.wsps = OB_WSPS_VALUE25;
```

```
ob_wwdgt0.weps = OB_WEPS_VALUE75;
```

```
ob_wwdgt0.ewie = OB_EWIE_ENABLE;
```

```
ob_wwdgt0.wdga = OB_WDGA_DISABLE;
```

```
state = ob_wwdgt0_write (ob_wwdgt0);
```

### ob\_wwdgt1\_wwdgt2\_write

The description of ob\_wwdgt1\_wwdgt2\_write is shown as below:

**Table 3-417. Function ob\_wwdgt1\_wwdgt2\_write**

<b>Function name</b>	ob_wwdgt1_wwdgt2_write
<b>Function prototype</b>	fmc_state_enum ob_wwdgt1_wwdgt2_write(uint16_t ob_wwdg_cnt)
<b>Function descriptions</b>	program option bytes WWDG1 and WWDG2
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_wwdg_cnt</b>	value in auto-start mode, 0 ~ 0x00003FFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

Example:

```
/* program option bytes WWDGT1 and WWDGT2 */
```

```
fmc_state_enum state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
state = ob_wwdgt1_wwdgt2_write(0x1F);
```

### ob\_fwdgt\_write

The description of ob\_fwdgt\_write is shown as below:

**Table 3-418. Function ob\_fwdgt\_write**

<b>Function name</b>	ob_fwdgt_write
<b>Function prototype</b>	fmc_state_enum ob_fwdgt_write(uint16_t ob_fwdgt_wnd, uint8_t ob_fwdgt_psc, uint8_t ob_fwdgt_rst, uint16_t ob_fwdgt_reload)
<b>Function descriptions</b>	program option bytes FWDGT0 and FWDGT1 and FWDGT2 and FWDGT3
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_fwdgt_wnd</b>	hardware free watchdog window value , 0 ~ 0x00000FFF
<b>Input parameter{in}</b>	
<b>ob_fwdgt_psc</b>	free watchdog timer prescaler selection
<b>OB_FWDGT_PSC4</b>	free watchdog timer prescaler selection 1/4

OB_FWDGT_PSC8	free watchdog timer prescaler selection 1/8
OB_FWDGT_PSC16	free watchdog timer prescaler selection 1/16
OB_FWDGT_PSC32	free watchdog timer prescaler selection 1/32
OB_FWDGT_PSC64	free watchdog timer prescaler selection 1/64
OB_FWDGT_PSC128	free watchdog timer prescaler selection 1/128
OB_FWDGT_PSC256	free watchdog timer prescaler selection 1/256
<b>Input parameter{in}</b>	
ob_fwdgt_rst	fwdgt to generate a reset
OB_FWDGT_RST_DISABLE	disable fwdgt to generate a reset
OB_FWDGT_RST_ENABLE	enable fwdgt to generate a reset
<b>Input parameter{in}</b>	
ob_fwdgt_reload	hardware free watchdog reload value , 0 ~ 0x00000FFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-382. fmc_state_enum</a>

```
/* program option bytes FWDGT */
```

```
fmc_state_enum state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
state = ob_fwdgt_write(0x1F, OB_FWDGT_PSC4, OB_FWDGT_RST_DISABLE, 0xFF);
```

### ob\_user\_get

The description of ob\_user\_get is shown as below:

**Table 3-419. Function ob\_user\_get**

<b>Function name</b>	ob_user_get
<b>Function prototype</b>	uint8_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>	
uint8_t	the FMC user option bytes values(0x0 – 0xFF)

Example:

```
/* get the FMC user option bytes */
```

```
uint8_t user;
```

```
user = ob_user_get();
```

### ob\_data\_get

The description of ob\_data\_get is shown as below:

**Table 3-420. Function ob\_data\_get**

<b>Function name</b>	ob_data_get
<b>Function prototype</b>	uint16_t ob_data_get(void);
<b>Function descriptions</b>	get the value of option bytes DATA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint16_t	the FMC data option bytes values(0x0 – 0xFFFF)

Example:

Example:

```
/* get the value of FMC option bytes OB_DATA in FMC_OBSTAT register */
```

```
uint16_t data = ob_data_get();
```

### ob\_write\_protection\_get

The description of ob\_write\_protection\_get is shown as below:

**Table 3-421. Function ob\_write\_protection\_get**

<b>Function name</b>	ob_write_protection_get
<b>Function prototype</b>	uint32_t ob_write_protection_get(void);
<b>Function descriptions</b>	get the value of option bytes write protection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	the FMC write protection option bytes value(0x0– 0xFFFFFFFF)

Example:

```
/* get the FMC option bytes write protection */
```

```
uint32_t wp;
```

```
wp = ob_write_protection_get();
```

### ob\_security\_protection\_flag\_get

The description of ob\_security\_protection\_flag\_get is shown as below:

**Table 3-422. Function ob\_security\_protection\_flag\_get**

<b>Function name</b>	ob_security_protection_flag_get
<b>Function prototype</b>	FlagStatus ob_security_protection_flag_get(void);
<b>Function descriptions</b>	get the FMC option bytes security protection state
<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 the FMC option bytes security protection */
```

```
FlagStatus flag;
```

```
flag = ob_security_protection_flag_get();
```

### fmc\_ecc\_error\_address\_get

The description of fmc\_ecc\_error\_address\_get is shown as below:

**Table 3-423. Function fmc\_ecc\_error\_address\_get**

<b>Function name</b>	fmc_ecc_error_address_get
<b>Function prototype</b>	uint16_t fmc_ecc_error_address_get(void)
<b>Function descriptions</b>	get the FMC ECC error address in register FMC_ECCCS
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>Uint16_t</b>	the FMC ECC error address (0x0 – 0x7FFF)

Example:



```
/* get the FMC data option bytes */
```

```
uint16_t ecc_addr;
```

```
ecc_addr = fmc_ecc_error_address_get();
```

### fmc\_flag\_get

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

**Table 3-424. Function fmc\_flag\_get**

<b>Function name</b>	fmc_flag_get
<b>Function prototype</b>	FlagStatus fmc_flag_get(fmc_flag_enum flag)
<b>Function descriptions</b>	get FMC flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	FMC flag, refer to <a href="#">Table 3-383. fmc_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get FMC end of operation flag */
```

```
FlagStatus flag;
```

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

### fmc\_flag\_clear

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

**Table 3-425. Function fmc\_flag\_clear**

<b>Function name</b>	fmc_flag_clear
<b>Function prototype</b>	void fmc_flag_clear(fmc_flag_enum 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, refer to <a href="#">Table 3-383. fmc_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear FMC program error flag */
```

```
fmc_flag_clear(FMC_FLAG_PGERR);
```

### fmc\_interrupt\_enable

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

**Table 3-426. Function fmc\_interrupt\_enable**

<b>Function name</b>	fmc_interrupt_enable
<b>Function prototype</b>	void fmc_interrupt_enable(fmc_interrupt_enum interrupt)
<b>Function descriptions</b>	enable FMC interrupt
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FMC interrupt source, refer to <a href="#">Table 3-385. fmc_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FMC end of operation interrupt */
```

```
fmc_unlock();
```

```
fmc_interrupt_enable(FMC_INT_END);
```

### fmc\_interrupt\_disable

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

**Table 3-427. Function fmc\_interrupt\_disable**

<b>Function name</b>	fmc_interrupt_disable
<b>Function prototype</b>	void fmc_interrupt_disable(fmc_interrupt_enum interrupt)
<b>Function descriptions</b>	disable FMC interrupt
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FMC interrupt source, refer to <a href="#">Table 3-385. fmc_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FMC end of operation interrupt */
```

```
fmc_unlock();
```

```
fmc_interrupt_disable(FMC_INT_END);
```

### fmc\_interrupt\_flag\_get

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

**Table 3-428. 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 int_flag)
<b>Function descriptions</b>	get FMC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	FMC interrupt flag, refer to <a href="#">Table 3-384. fmc_interrupt_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get FMC operation error interrupt flag bit */
```

```
FlagStatus flag;
```

```
flag = fmc_interrupt_flag_get (FMC_INT_FLAG_PGERR);
```

### fmc\_interrupt\_flag\_clear

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

**Table 3-429. Function fmc\_interrupt\_flag\_clear**

<b>Function name</b>	fmc_interrupt_flag_clear
<b>Function prototype</b>	void fmc_interrupt_flag_clear(fmc_interrupt_flag_enum int_flag)
<b>Function descriptions</b>	clear FMC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	FMC interrupt flag, refer to <a href="#">Table 3-384. fmc_interrupt_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear FMC operation error interrupt flag */
```

```
fmc_interrupt_flag_clear (FMC_INT_FLAG_PGERR);
```

## 3.15. 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 [3.15.1](#). The FWDGT firmware functions are introduced in chapter [3.15.2](#).

### 3.15.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

**Table 3-430. FWDGT Registers**

Registers	Descriptions
FWDGT_CTL	control register
FWDGT_PSC	prescaler register
FWDGT_RLD	reload register
FWDGT_STAT	status register
FWDGT_WND	window register
FWDGT_RCR	reset control register

### 3.15.2. Descriptions of Peripheral functions

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

**Table 3-431. 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_reset_output	configure the FWDGT reset output
fwdgt_interrupt_output	configure the FWDGT interrupt output
fwdgt_flag_get	get flag state of FWDGT

## fwdgt\_write\_enable

The description of fwdgt\_write\_enable is shown as below:

**Table 3-432. 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 3-433. 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 3-434. Function fwdgt\_enable**

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

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 3-435. Function fwdgt\_prescaler\_value\_config**

<b>Function name</b>	fwdgt_prescaler_value_config
<b>Function prototype</b>	ErrStatus fwdgt_prescaler_value_config(uint16_t prescaler_value);
<b>Function descriptions</b>	configure the FWDGT counter clock prescaler value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>prescaler_value</b>	specify prescaler value
<i>FWDGT_PSC_DIVx</i>	FWDGT prescaler set to x(x=4, 8, 16, 32, 64, 128, 256)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	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 3-436. Function fwdgt\_reload\_value\_config

Function name	fwdgt_reload_value_config
Function prototype	ErrStatus fwdgt_reload_value_config(uint16_t reload_value);
Function descriptions	configure the FWDGT counter reload value
Precondition	-
The called functions	-
Input parameter{in}	
reload_value	reload_value, specify reload value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

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

```
ErrStatus flag;
```

```
flag = fwdgt_reload_value_config(0xFFF);
```

### fwdgt\_window\_value\_config

The description of fwdgt\_window\_value\_config is shown as below:

Table 3-437. Function fwdgt\_window\_value\_config

Function name	fwdgt_window_value_config
Function prototype	ErrStatus fwdgt_window_value_config(uint16_t window_value);
Function descriptions	configure the FWDGT counter window value
Precondition	-
The called functions	-
Input parameter{in}	
window_value	window_value, specify window value(0x0000 - 0x0FFF)
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR / SUCCESS

Example:

```
/* set FWDGT window value to 0xFFF */
```

```
ErrStatus flag;
```

```
flag = fwdgt_window_value_config(0xFFF);
```

### fwdgt\_counter\_reload

The description of fwdgt\_counter\_reload is shown as below:

Table 3-438. Function `fwdgt_counter_reload`

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

Example:

```
/* reload FWDGT counter */
```

```
fwdgt_counter_reload();
```

### `fwdgt_config`

The description of `fwdgt_config` is shown as below:

Table 3-439. Function `fwdgt_config`

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

Example:



```
/* configure FWDGT counter clock: 32KHz(IRC32K) / 64 = 0.5 KHz */
```

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

### fwdgt\_reset\_output

The description of fwdgt\_reset\_output is shown as below:

**Table 3-440. Function fwdgt\_reset\_output**

<b>Function name</b>	fwdgt_reset_output
<b>Function prototype</b>	void fwdgt_reset_output(void);
<b>Function descriptions</b>	configure the FWDGT reset output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the FWDGT reset output */
```

```
void fwdgt_reset_output(void);
```

### fwdgt\_interrupt\_output

The description of fwdgt\_interrupt\_output is shown as below:

**Table 3-441. Function fwdgt\_interrupt\_output**

<b>Function name</b>	fwdgt_interrupt_output
<b>Function prototype</b>	void fwdgt_interrupt_output(void);
<b>Function descriptions</b>	configure the FWDGT interrupt output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the FWDGT interrupt output */
```

```
void fwdgt_interrupt_output(void);
```

## fwdgt\_flag\_get

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

**Table 3-442. Function fwdgt\_flag\_get**

<b>Function name</b>	fwdgt_flag_get
<b>Function prototype</b>	FlagStatus fwdgt_flag_get(uint16_t flag);
<b>Function descriptions</b>	get flag state of FWDGT
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	flag to get
<i>FWDGT_FLAG_PUD</i>	a write operation to FWDGT_PSC register is on going
<i>FWDGT_FLAG_RUD</i>	a write operation to FWDGT_RLD register is on going
<i>FWDGT_FLAG_WUD</i>	a write operation to FWDGT_WND register is on going
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* test if a prescaler value update is on going */
```

```
FlagStatus status;
```

```
status = fwdgt_flag_get(FWDGT_FLAG_PUD);
```

## 3.16. GPIO

GPIO is used to implement logic input/output functions for the devices. The GPIO registers are listed in chapter [3.16.1](#), the GPIO firmware functions are introduced in chapter [3.16.2](#).

### 3.16.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

**Table 3-1. GPIO Registers**

Registers	Descriptions
GPIOx_CTL	GPIO port control register
GPIOx_OMODE	GPIO port output mode register
GPIOx_OSPD	GPIO port output speed register
GPIOx_PUD	GPIO port pull-up/pull-down register
GPIOx_ISTAT	GPIO port input status register
GPIOx_OCTL	GPIO port output control register
GPIOx_BOP	GPIO port bit operation register

GPIOx_LOCK	GPIO port configuration lock register
GPIOx_AFSEL0	GPIO alternate function selected register 0
GPIOx_AFSEL1	GPIO alternate function selected register 1
GPIOx_BC	GPIO bit clear register
GPIOx_TG	GPIO port bit toggle register
GPIOx_HLD	GPIO port hold register
GPIOx_FUNCSEL	GPIO port bit select extension register
GPIOx_HLDCTL	GPIO port global hold register

### 3.16.2. Descriptions of Peripheral functions

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

**Table 3-2. GPIO firmware function**

Function name	Function description
gpio_deinit	reset GPIO port
gpio_mode_set	set GPIO mode
gpio_output_options_set	set GPIO output type and speed
gpio_bit_set	set GPIO pin bit
gpio_bit_reset	reset GPIO pin bit
gpio_bit_write	write data to the specified GPIO pin
gpio_port_write	write data to the specified GPIO port
gpio_input_bit_get	get GPIO pin input status
gpio_input_port_get	get GPIO port input status
gpio_output_bit_get	get GPIO pin output status
gpio_output_port_get	get GPIO port output status
gpio_af_set	set GPIO alternate function
gpio_pin_lock	lock GPIO pin bit
gpio_bit_toggle	toggle GPIO pin status
gpio_port_toggle	toggle GPIO port status
gpio_bit_hold_enable	enable hold GPIO pin data output
gpio_bit_hold_disable	disable hold GPIO pin data output
gpio_tsel_function_enable	enable tsel function
gpio_tsel_function_disable	disable tsel function
gpio_one_line_function_enable	enable one line function
gpio_one_line_function_disable	disable one line function
gpio_sample_hold_enable	enable data output function

#### gpio\_deinit

The description of gpio\_deinit is shown as below:

**Table 3-3. Function gpio\_deinit**

Function name	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,N)
<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 3-4. 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,N)
<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_PULLDOWN</i>	with pull-down resistor

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	
-	-

Exmple:

```
/* 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\_mode\_set is shown as below:

**Table 3-5. 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>	-
Input parameter{in}	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,N)
Input parameter{in}	
<b>otype</b>	gpio pin output mode
<i>GPIO_OTYPE_PP</i>	push pull mode
<i>GPIO_OTYPE_OD</i>	open drain mode
Input parameter{in}	
<b>speed</b>	gpio pin output max speed
<i>GPIO_OSPEED_LOW</i>	output max speed low drive
<i>GPIO_OSPEED_HIGH</i>	output max speed high drive
<i>GPIO_OSPEED_LARGE_CURRENT</i>	output max speed high current drive
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	

-	-
---	---

Exmple:

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

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

### gpio\_bit\_set

The description of gpio\_bit\_set is shown as below:

**Table 3-6. 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>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,N)
<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>	
-	-

Exmple:

```
/* set PA0 */

gpio_bit_set(GPIOA, GPIO_PIN_0);
```

### gpio\_bit\_reset

The description of gpio\_bit\_reset is shown as below:

**Table 3-7. 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,N)
<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_set(GPIOA, GPIO_PIN_0);
```

### gpio\_bit\_write

The description of gpio\_bit\_write is shown as below:

**Table 3-8. 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,N)
<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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-443. Function gpio\_port\_write**

<b>Function name</b>	gpio_port_write
<b>Function prototype</b>	void gpio_port_write(uint32_t gpio_periph,uint16_t data);
<b>Function descriptions</b>	write data to the specified GPIO port
<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,N)
<b>Input parameter{in}</b>	
<b>data</b>	specify the value to be written to the port output data register
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

## gpio\_input\_bit\_get

The description of gpio\_input\_bit\_get is shown as below:

**Table 3-444. Function gpio\_input\_bit\_get**

<b>Function name</b>	gpio_input_bit_get
<b>Function prototype</b>	FlagStatus gpio_input_bit_get(uint32_t gpio_periph,uint32_t pin);
<b>Function descriptions</b>	get GPIO pin 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,N)
<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>	
-	-



Return value	
FlagStatus	SET / RESET

Example:

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

```
FlagStatus bit_state;
```

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

### gpio\_input\_port\_get

The description of gpio\_input\_port\_get is shown as below:

**Table 3-445. Function gpio\_input\_port\_get**

Function name	gpio_input_port_get
Function prototype	uint16_t gpio_input_port_get(uint32_t gpio_periph);
Function descriptions	get GPIO all pins input status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G,N)
Output parameter{out}	
-	-
Return value	
uint16_t	0x0000-0xFFFF

Example:

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

```
uint16_t port_state;
```

```
port_state = gpio_input_port_get(GPIOA);
```

### gpio\_output\_bit\_get

The description of gpio\_output\_bit\_get is shown as below:

**Table 3-446. Function gpio\_output\_bit\_get**

Function name	gpio_output_bit_get
Function prototype	FlagStatus gpio_output_bit_get(uint32_t gpio_periph,uint32_t pin);
Function descriptions	get GPIO pin output status
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port

<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,N)
<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 3-447. 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 all pins 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,N)
<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 3-448. 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
GPIOx	GPIOx(x = A,B,C,D,E,F,G,N)
<b>Input parameter{in}</b>	
<b>alt_func_num</b>	GPIO pin af function, please refer to specific device datasheet
GPIO_AF_0	CK_OUT, TRACECK, TRACED0, TRACED1, TRACED2, TRACED3, CPTIMERW
GPIO_AF_1	TIMER0, TIMER1, TIMER2, TIMER7
GPIO_AF_2	TIMER_ETI, TIMER0, TIMER7
GPIO_AF_3	TIMER0, TIMER2, TIMER7
GPIO_AF_4	TIMER0, TIMER7, POC
GPIO_AF_5	CFMUREF, ADC0, ADC2, ADCSM1, ADCSM2
GPIO_AF_6	UART0, UART1, UART2, UART3, POC
GPIO_AF_7	UART3, ADC0, ADC2, TIMER0, TIMER7, I2C
GPIO_AF_8	CMP0, CMP1, CMP2, CMP3, I2C, CAN
GPIO_AF_9	TIMER2, SPI
GPIO_AF_10	GTOC0, GPTIMER0, GPTIMER1, UART3, ADCSM3, ADCSM4
GPIO_AF_11	GTOC0, GTOC1, SPI, GPTIMER0, GPTIMER1
GPIO_AF_12	GTOC2, GPTIMER0, GPTIMER1
GPIO_AF_13	GTOC2, GTOC3, CAN, GPTIMER0, GPTIMER1
GPIO_AF_14	GPTIMER0, GPTIMER1
GPIO_AF_15	EVENT_OUT
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
GPIO_PIN_x	GPIO_PIN_x(x=0..15)
GPIO_PIN_ALL	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 3-449. 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,N)
<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 3-450. 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,N)
<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>	GPIO_PIN_ALL
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

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

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

### gpio\_port\_toggle

The description of gpio\_port\_toggle is shown as below:

**Table 3-451. 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,N)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* toggle GPIOA */
```

```
gpio_port_toggle(GPIOA);
```

### gpio\_bit\_hold\_enable

The description of gpio\_bit\_hold\_enable is shown as below:

**Table 3-9. Function gpio\_bit\_hold\_enable**

<b>Function name</b>	gpio_bit_hold_enable
<b>Function prototype</b>	void gpio_bit_hold_enable(uint32_t gpio_periph, uint32_t pin)
<b>Function descriptions</b>	enable hold GPIO pin data output
<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,N)
<i>GPIO pin</i>	GPIO_PIN_x(x =0..15), GPIO_PIN_ALL

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* toggle GPIOA */
```

```
gpio_bit_hold_disable (GPIOA);
```

### gpio\_bit\_hold\_disable

The description of gpio\_bit\_hold\_disable is shown as below:

**Table 3-10. Function gpio\_bit\_hold\_disable**

Function name	gpio_bit_hold_disable
Function prototype	void gpio_bit_hold_disable(uint32_t gpio_periph, uint32_t pin)
Function descriptions	disable hold GPIO pin data output
Precondition	-
The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G,N)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* toggle GPIOA */
```

```
gpio_bit_hold_disable (GPIOA);
```

### gpio\_tsel\_function\_enable

The description of gpio\_tsel\_function\_enable is shown as below:

**Table 3-11. Function gpio\_tsel\_function\_enable**

Function name	gpio_tsel_function_enable
Function prototype	void gpio_tsel_function_enable(void)
Function descriptions	enable tsel function
Precondition	-
The called functions	-
Input parameter{in}	
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* enable tsel function */
```

```
gpio_tsel_function_enable (void);
```

### gpio\_tsel\_function\_disable

The description of gpio\_tsel\_function\_disable is shown as below:

**Table 3-12. Function gpio\_tsel\_function\_disable**

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

Example:

```
/* disable tsel function */
```

```
gpio_tsel_function_disable (void);
```

### gpio\_one\_line\_function\_enable

The description of gpio\_one\_line\_function\_enable is shown as below:

**Table 3-13. Function gpio\_one\_line\_function\_enable**

<b>Function name</b>	gpio_one_line_function_enable
<b>Function prototype</b>	void gpio_one_line_function_enable(void)
<b>Function descriptions</b>	enable one line function
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable one line function */

gpio_one_line_function_enable (void);
```

### gpio\_one\_line\_function\_disable

The description of gpio\_one\_line\_function\_disable is shown as below:

**Table 3-14. Function gpio\_one\_line\_function\_disable**

<b>Function name</b>	gpio_one_line_function_disable
<b>Function prototype</b>	void gpio_one_line_function_disable (void)
<b>Function descriptions</b>	disable one line 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 one line function */

gpio_one_line_function_disable (void);
```

### gpio\_sample\_hold\_enable

The description of gpio\_sample\_hold\_enable is shown as below:

**Table 3-15. Function gpio\_sample\_hold\_enable**

<b>Function name</b>	gpio_sample_hold_enable
<b>Function prototype</b>	void gpio_sample_hold_enable (void)
<b>Function descriptions</b>	enable data output 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 data output function */
```

```
gpio_sample_hold_enable(void);
```

## 3.17. GPTIMER

The general purpose timer module (GPTIMER0/1) is two-channel timer that supports both input capture and output compare. The general purpose timer has a 16-bit counter that can be used as an unsigned counter. The GPTIMER registers are listed in chapter [3.17.1](#), the GPTIMER firmware functions are introduced in chapter [3.17.2](#).

### 3.17.1. Descriptions of Peripheral registers

GPTIMER registers are listed in the table shown as below:

**Table 3-452. GPTIMER registers**

Function name	Function description
GPTIMER_WP	GPTIMER write protect register
GPTIMER_SWEN	GPTIMER software enable register
GPTIMER_SWDIS	GPTIMER software disable register
GPTIMER_SWRST	GPTIMER software reset register
GPTIMER_ESSEL	GPTIMER counter enable source selection register
GPTIMER_DSSEL	GPTIMER counter disable source selection register
GPTIMER_RSSEL	GPTIMER counter reset source selection register
GPTIMER_CH0CSSEL	GPTIMER channel 0 capture source selection register
GPTIMER_CH1CSSEL	GPTIMER channel 1 capture source selection register
GPTIMER_CTL0	GPTIMER control register 0
GPTIMER_CUPEVSSEL L	GPTIMER count up event source selection register
GPTIMER_CDNEVSSEL L	GPTIMER count down event source selection register
GPTIMER_CH0CTL	GPTIMER channel 0 control register
GPTIMER_CH1CTL	GPTIMER channel 1 control register
GPTIMER_CHCTL	GPTIMER channel control register
GPTIMER_DMAINTEN	GPTIMER dma and interrupt enable register
GPTIMER_INTF	GPTIMER interrupt flag register
GPTIMER_UPSSEL	GPTIMER update source selection register
GPTIMER_CNT	GPTIMER counter register
GPTIMER_PSC	GPTIMER prescaler register
GPTIMER_CAR	GPTIMER counter auto reload register
GPTIMER_CH0CV	GPTIMER channel 0 capture/compare value register
GPTIMER_CH1CV	GPTIMER channel 1 capture/compare value register
GPTIMER_CH0COMV_ ADD	GPTIMER channel 0 additional compare value register

GPTIMER_CH1COMV_ADD	GPTIMER channel 1 additional compare value register
GPTIMER_DTCTL	GPTIMER dead-time control register
GPTIMER_ADCTL	GPTIMER ADC trigger control register
GPTIMER_ADCCR1	GPTIMER ADC trigger compare value 1 register
GPTIMER_ADCCR2	GPTIMER ADC trigger compare value 2 register
GPTIMER_ADCTRGS	GPTIMER ADC trigger skipping register
GPTIMER_ADDINTSC TL0	GPTIMER additional interrupt skipping control register 0
GPTIMER_ADDINTSC TL1	GPTIMER additional interrupt skipping control register 1
GPTIMER_IADCTSS	GPTIMER interrupt and ADC trigger signal skipping register
GPTIMER_CREP	GPTIMER counter repetition register
GPTIMER_SYN_RSTCTL	GPTIMER synchronous reset control register
GPTIMER_DMCFG	GPTIMER DMA configuration register
GPTIMER_DMATB	GPTIMER DMA transfer buffer register
GPTIMER_CFG	GPTIMER configuration register

### 3.17.2. Descriptions of Peripheral functions

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

**Table 3-453. GPTIMER firmware function**

Function name	Function description
<code>gptimer_deinit</code>	deinit GPTIMER
<code>gptimer_struct_para_init</code>	initialize GPTIMER init parameter struct with a default value
<code>gptimer_init</code>	initialize GPTIMER counter
<code>gptimer_register_write_protect_enable</code>	enable GPTIMER register write protect function
<code>gptimer_register_write_protect_disable</code>	disable GPTIMER register write protect function
<code>gptimer_clock_source_select</code>	select a GPTIMER clock source
<code>gptimer_clock_polarity_config</code>	configure a GPTIMER clock polarity
<code>gptimer_counter_software_enable</code>	software enable GPTIMER counter
<code>gptimer_counter_software_disable</code>	software disable GPTIMER counter
<code>gptimer_counter_software_reset</code>	software reset GPTIMER counter
<code>gptimer_counter_enable_source_init</code>	initialize GPTIMER counter enable source init parameter struct with a default value

nit	
gptimer_counter_enable_source_config	initialize GPTIMER counter enable source
gptimer_counter_disable_source_struct_parameter_init	initialize GPTIMER counter disable source init parameter struct with a default value
gptimer_counter_disable_source_config	initialize GPTIMER counter disable source
gptimer_counter_reset_source_struct_parameter_init	initialize GPTIMER counter reset source init parameter struct with a default value
gptimer_counter_reset_source_config	initialize GPTIMER counter reset source
gptimer_counter_up_source_struct_parameter_init	initialize GPTIMER counter up source init parameter struct with a default value
gptimer_counter_up_source_config	initialize GPTIMER counter up source
gptimer_counter_down_source_struct_parameter_init	initialize GPTIMER counter down source init parameter struct with a default value
gptimer_counter_down_source_config	initialize GPTIMER counter down source
gptimer_counter_up_input_level_source_select	select counter up input level source
gptimer_counter_down_input_level_source_select	select counter down input level source
gptimer_enable	enable GPTIMER
gptimer_disable	disable GPTIMER
gptimer_auto_reload_shadow_enable	enable the auto reload shadow function
gptimer_auto_reload_shadow_disable	disable the auto reload shadow function
gptimer_update_event_enable	enable the update event
gptimer_update_event_disable	disable the update event
gptimer_counter_alignment	set counter alignment mode
gptimer_counter_up_direction	set counter up direction
gptimer_counter_down_direction	set counter down direction
gptimer_counter_direction_force_set_enable	enable counter direction force set

gptimer_counter_directi on_force_set_disable	disable counter direction force set
gptimer_register_global _update_source_select	select register global update source
gptimer_register_local_ update_source_select	select register local update source
gptimer_prescaler_conf g	configure prescaler
gptimer_update_repetiti on_value_config	configure update repetition register value
gptimer_update_repetiti on_counter_read	read update repetition register value
gptimer_autoreload_val ue_config	configure autoreload register value
gptimer_counter_value_ config	configure counter register value
gptimer_counter_read	read counter value
gptimer_prescaler_read	read prescaler value
gptimer_single_pulse_ mode_config	configure single pulse mode
gptimer_dma_enable	enable DMA
gptimer_dma_disable	disable DMA
gptimer_dma_transfer_ config	configure the DMA transfer
gptimer_channel_output _struct_para_init	initialize GPTIMER channel output parameter struct with a default value
gptimer_channel_output _config	configure channel output mode
gptimer_channel_output _force_duty_config	configure channel output force duty
gptimer_channel_output _compare_value_config	configure channel output compare value
gptimer_channel_output _additional_compare_v alue_config	configure channel output additional compare value
gptimer_channel_output _shadow_config	configure channel output shadow function
gptimer_channel_output _additional_shadow_co nfig	configure channel output additional shadow function
gptimer_channel_output _control_shadow_confi g	configure channel output additional shadow function

gptimer_two_channel_output_high_check_enable	enable two channel simultaneous output high check function
gptimer_two_channel_output_high_check_disable	disable two channel simultaneous output high check function
gptimer_two_channel_output_low_check_enable	enable two channel simultaneous output low check function
gptimer_two_channel_output_low_check_disable	disable two channel simultaneous output low check function
gptimer_period_signal_output_enable	enable counter cycle synchronization signal output
gptimer_period_signal_output_disable	disable counter cycle synchronization signal output
gptimer_stop_output_set_select	configure stop output source
gptimer_stop_output_recover_time_select	output stop recover time select
gptimer_channel_complementary_output_struct_para_init	initialize GPTIMER complementary output parameter struct with a default value
gptimer_channel_complementary_output_config	configure GPTIMER channel complementary output function
gptimer_channel_io_direction_config	configure channel input or output
gptimer_channel_io_state_config	configure channel input or output enable state
gptimer_capture_source_struct_para_init	initialize GPTIMER counter capture source init parameter struct with a default value
gptimer_capture_source_config	initialize GPTIMER counter capture source
gptimer_channel_input_capture_filter_config	configure channel input capture filter
gptimer_channel_input_capture_prescaler_config	configure channel input capture prescaler
gptimer_channel_capture_value_register_read	read channel capture compare register value
gptimer_counter_sync_control_select	select counter sync reset set
gptimer_counter_sync_reset_source_select	select counter sync reset source

gptimer_trigger_adc_compare_enable	enable compare event produce a ADC converter trigger signal
gptimer_trigger_adc_compare_disable	disable compare event produce a ADC converter trigger signal
gptimer_trigger_adc_compare_value_config	config trigger adc compare register value
gptimer_trigger_adc_compare_shadow_enable	enable adc compare register shadow function
gptimer_trigger_adc_compare_shadow_disable	disable adc compare register shadow function
gptimer_trigger_adc_adsm_enable	enable trigger adc adsm signal
gptimer_trigger_adc_adsm_disable	disable trigger adc adsm signal
gptimer_trigger_adc_adsm_select	select trigger adc adsm signal
gptimer_trigger_adc_skipping_config	config trigger adc skipping value
gptimer_trigger_adc_skipping_time_select	select trigger adc skipping function
gptimer_trigger_adc_skipping_counter_read	read trigger adc skipping counter
gptimer_additional_interrupt_skipping_config	config additional interrupt skipping initial value
gptimer_additional_interrupt_skipping_time_select	select additional interrupt skipping function
gptimer_additional_interrupt_skipping_counter_read	read additional interrupt skipping counter
gptimer_flow_interrupt_skipping_source_select	select flow interrupt skipping source
gptimer_flow_interrupt_skipping_link_enable	enable flow interrupt skipping link function
gptimer_flow_interrupt_skipping_link_disable	disable flow interrupt skipping link function
gptimer_flow_interrupt_skipping_num_config	configure flow interrupt repetition skipping number
gptimer_flow_interrupt_skipping_counter_read	read flow interrupt skipping counter value
gptimer_write_chxval_register_config	configure write CHxVAL register selection
gptimer_flag_get	get GPTIMER flags

gptimer_flag_clear	reset GPTIMER flags
gptimer_interrupt_enable	enable the GPTIMER interrupt
gptimer_interrupt_disable	disable the GPTIMER interrupt
gptimer_interrupt_flag_get	get GPTIMER interrupt flags
gptimer_interrupt_flag_clear	clear GPTIMER interrupt flags

### Structure gptimer\_parameter\_struct

**Table 3-454. Structure gptimer\_parameter\_struct**

Member name	Function description
clock_source	clock source
clock_polarity	clock polarity
prescaler	prescaler value
alignedmode	aligned mode
counterdirection	counter direction
period	period value
clockdivision	clock division value
repetitioncounter	the counter repetition value

### Structure gptimer\_counter\_enable\_source\_parameter\_struct

**Table 3-455. Structure gptimer\_counter\_enable\_source\_parameter\_struct**

Member name	Function description
enable_source_eti0	eti0 as counter enable source
enable_source_eti1	eti1 as counter enable source
enable_source_eti2	eti2 as counter enable source
enable_source_eti3	eti3 as counter enable source
enable_source_ch0	channel 0 as counter enable source
enable_source_ch1	channel 1 as counter enable source
enable_source_evsel0	evsel0 as counter enable source
enable_source_evsel1	evsel1 as counter enable source
enable_source_evsel2	evsel2 as counter enable source
enable_source_evsel3	evsel3 as counter enable source
enable_source_evsel4	evsel4 as counter enable source
enable_source_evsel5	evsel5 as counter enable source
enable_source_evsel6	evsel6 as counter enable source
enable_source_evsel7	evsel7 as counter enable source
enable_source_software	software event as counter enable source

## Structure gptimer\_counter\_disable\_source\_parameter\_struct

**Table 3-456. Structure gptimer\_counter\_disable\_source\_parameter\_struct**

Member name	Function description
disable_source_eti0	eti0 as counter disable source
disable_source_eti1	eti1 as counter disable source
disable_source_eti2	eti2 as counter disable source
disable_source_eti3	eti3 as counter disable source
disable_source_ch0	channel 0 as counter disable source
disable_source_ch1	channel 1 as counter disable source
disable_source_evsel0	evsel0 as counter disable source
disable_source_evsel1	evsel1 as counter disable source
disable_source_evsel2	evsel2 as counter disable source
disable_source_evsel3	evsel3 as counter disable source
disable_source_evsel4	evsel4 as counter disable source
disable_source_evsel5	evsel5 as counter disable source
disable_source_evsel6	evsel6 as counter disable source
disable_source_evsel7	evsel7 as counter disable source
disable_source_software	software event as counter disable source

## Structure gptimer\_counter\_reset\_source\_parameter\_struct

**Table 3-457. Structure gptimer\_counter\_reset\_source\_parameter\_struct**

Member name	Function description
reset_source_eti0	eti0 as counter reset source
reset_source_eti1	eti1 as counter reset source
reset_source_eti2	eti2 as counter reset source
reset_source_eti3	eti3 as counter reset source
reset_source_ch0	channel 0 as counter reset source
reset_source_ch1	channel 1 as counter reset source
reset_source_evsel0	evsel0 as counter reset source
reset_source_evsel1	evsel1 as counter reset source
reset_source_evsel2	evsel2 as counter reset source
reset_source_evsel3	evsel3 as counter reset source
reset_source_evsel4	evsel4 as counter reset source
reset_source_evsel5	evsel5 as counter reset source
reset_source_evsel6	evsel6 as counter reset source
reset_source_evsel7	elcn as counter reset source
reset_source_com_cap_sync	copmare, capture or synchronous reset event as counter reset source
reset_source_software	software event counter reset source



## Structure gptimer\_capture\_source\_parameter\_struct

**Table 3-458. Structure gptimer\_capture\_source\_parameter\_struct**

Member name	Function description
capture_source_eti0	eti0 as capture source
capture_source_eti1	eti1 as capture source
capture_source_eti2	eti2 as capture source
capture_source_eti3	eti3 as capture source
capture_source_ch0	channel 0 as capture source
capture_source_ch1	channel 1 as capture source
capture_source_evsel0	evsel0 as capture source
capture_source_evsel1	evsel1 as capture source
capture_source_evsel2	evsel2 as capture source
capture_source_evsel3	evsel3 as capture source
capture_source_evsel4	evsel4 as capture source
capture_source_evsel5	evsel5 as capture source
capture_source_evsel6	evsel6 as capture source
capture_source_evsel7	evsel7 as capture source

## Structure gptimer\_counter\_up\_source\_parameter\_struct

**Table 3-459. Structure gptimer\_counter\_up\_source\_parameter\_struct**

Member name	Function description
up_count_source_eti0	eti0 as up count source
up_count_source_eti1	eti1 as up count source
up_count_source_eti2	eti2 as up count source
up_count_source_eti3	eti3 as up count source
up_count_source_ch0	channel 0 as up count source
up_count_source_ch1	channel 1 as up count source
up_count_source_evsel 0	evsel0 as up count source
up_count_source_evsel 1	evsel1 as up count source
up_count_source_evsel 2	evsel2 as up count source
up_count_source_evsel 3	evsel3 as up count source
up_count_source_evsel 4	evsel4 as up count source
up_count_source_evsel 5	evsel5 as up count source
up_count_source_evsel 6	evsel6 as up count source
up_count_source_evsel	evsel7 as up count source

### Structure gptimer\_counter\_down\_source\_parameter\_struct

**Table 3-460. Structure gptimer\_counter\_down\_source\_parameter\_struct**

Member name	Function description
down_count_source_eti 0	eti0 as down count source
down_count_source_eti 1	eti1 as down count source
down_count_source_eti 2	eti2 as down count source
down_count_source_eti 3	eti3 as down count source
down_count_source_ch 0	channel 0 as down count source
down_count_source_ch 1	channel 1 as down count source
down_count_source_ev sel0	evsel0 as down count source
down_count_source_ev sel1	evsel1 as down count source
down_count_source_ev sel2	evsel2 as down count source
down_count_source_ev sel3	evsel3 as down count source
down_count_source_ev sel4	evsel4 as down count source
down_count_source_ev sel5	evsel5 as down count source
down_count_source_ev sel6	evsel6 as down count source
down_count_source_ev sel7	evsel7 as down count source

### Structure gptimer\_oc\_parameter\_struct

**Table 3-461. Structure gptimer\_oc\_parameter\_struct**

Member name	Function description
chxcv_up_output_level	up count chxcv match output level
chxcv_down_output_level	down count chxcv match output level
chxcomv_add_up_output_level	up count chxcomv_add match output level

chxcomv_add_down_output_level	down count chxcomv_add match output level
period_end_output_level	period end output level
cnt_enable_disable_output_control_enable	counter start or stop output control enable
cnt_enable_output_initial_level	counter enable output initial level
cnt_disable_output_level	counter disable output level
output_stop_output_level	output stop output level
force_duty_output_mode	force duty output mode
force_duty_output_time	force duty output time
force_duty_end_output_level	force duty end output level
compare_period_end_priority_control	compare match and period end priority control

### Structure gptimer\_com\_oc\_parameter\_struct

**Table 3-462. Structure gptimer\_com\_oc\_parameter\_struct**

Member name	Function description
complementary_mode	complementary output mode
deadtime_enable	deadtime enable
deadtime_mode	deadtime mode
rising_deadtime_value	rising edge deadtime value
falling_deadtime_value	falling edge deadtime value
rising_deadtime_shadow	rising edge deadtime shadow
falling_deadtime_shadow	falling edge deadtime shadow

### gptimer\_deinit

The description of gptimer\_deinit is shown as below:

**Table 3-463. Function gptimer\_deinit**

Function name	gptimer_deinit
Function prototype	void gptimer_deinit(uint32_t timer_periph)
Function descriptions	deinit GPTIMER
Precondition	-
The called functions	rcu_periph_reset_enable(),rcu_periph_reset_disable()
Input parameter{in}	

<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinit GPTIMER0 */
```

```
gptimer_deinit(GPTIMER0);
```

### **gptimer\_struct\_para\_init**

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

**Table 3-464. Function gptimer\_struct\_para\_init**

<b>Function name</b>	gptimer_struct_para_init
<b>Function prototype</b>	void gptimer_struct_para_init(gptimer_parameter_struct *initpara)
<b>Function descriptions</b>	initialize GPTIMER init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>initpara</b>	refer to <a href="#">Table 3-454. Structure gptimer_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* init parameter struct with a default value */
```

```
gptimer_parameter_struct para;
```

```
gptimer_struct_para_init(&para);
```

### **gptimer\_init**

The description of gptimer\_init is shown as below:

**Table 3-465. Function gptimer\_init**

<b>Function name</b>	gptimer_init
<b>Function prototype</b>	void gptimer_init(uint32_t timer_periph, gptimer_parameter_struct *initpara)
<b>Function descriptions</b>	initialize GPTIMER counter
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
Input parameter{in}	
<b>initpara</b>	refer to <a href="#">Table 3-454. Structure gptimer_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize GPTIMER0 counter */
```

```
gptimer_parameter_struct para;
```

```
gptimer_init(GPTIMER0, para);
```

### gptimer\_register\_write\_protect\_enable

The description of gptimer\_register\_write\_protect\_enable is shown as below:

**Table 3-466. Function gptimer\_register\_write\_protect\_enable**

<b>Function name</b>	gptimer_register_write_protect_enable
<b>Function prototype</b>	void gptimer_register_write_protect_enable(uint32_t timer_periph, uint32_t register_protect)
<b>Function descriptions</b>	enable GPTIMER register write protect function
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
Input parameter{in}	
<b>register_protect</b>	register write protect mode
<i>GPTIMER_WRITE_PROTECT_SWEN</i>	GPTIMER_SWEN register writer protect
<i>GPTIMER_WRITE_PROTECT_SWDIS</i>	GPTIMER_SWDIS register writer protect
<i>GPTIMER_WRITE_PROTECT_SWRST</i>	GPTIMER_SWRST register writer protect
<i>GPTIMER_WRITE_PROTECT_OTHER</i>	other register writer protect
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable GPTIMER0 register write protect function */

gptimer_register_write_protect_enable(GPTIMER0,
GPTIMER_WRITE_PROTECT_OTHER);
```

### **gptimer\_register\_write\_protect\_disable**

The description of gptimer\_register\_write\_protect\_disable is shown as below:

**Table 3-467. Function gptimer\_register\_write\_protect\_disable**

<b>Function name</b>	gptimer_register_write_protect_disable
<b>Function prototype</b>	void gptimer_register_write_protect_disable(uint32_t timer_periph, uint32_t register_protect)
<b>Function descriptions</b>	disable GPTIMER register write protect function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>register_protect</b>	register write protect mode
<i>GPTIMER_WRITE_PROTECT_SWEN</i>	GPTIMER_SWEN register writer protect
<i>GPTIMER_WRITE_PROTECT_SWDIS</i>	GPTIMER_SWDIS register writer protect
<i>GPTIMER_WRITE_PROTECT_SWRST</i>	GPTIMER_SWRST register writer protect
<i>GPTIMER_WRITE_PROTECT_OTHER</i>	other register writer protect
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 register write protect function */

gptimer_register_write_protect_disable(GPTIMER0,
GPTIMER_WRITE_PROTECT_OTHER);
```

### **gptimer\_clock\_source\_select**

The description of gptimer\_clock\_source\_select is shown as below:

Table 3-468. Function `gptimer_clock_source_select`

<b>Function name</b>	<code>gptimer_clock_source_select</code>
<b>Function prototype</b>	<code>void gptimer_clock_source_select(uint32_t timer_periph, uint32_t clock_source)</code>
<b>Function descriptions</b>	select a GPTIMER clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>clock_source</b>	clock source select
<i>GPTIMER_CLOCK_SOURCE_CK_GPTIMER</i>	select CK_GPTIMER as clock source
<i>GPTIMER_CLOCK_SOURCE_ETI0</i>	select ETI0 as clock source
<i>GPTIMER_CLOCK_SOURCE_ETI1</i>	select ETI1 as clock source
<i>GPTIMER_CLOCK_SOURCE_ETI2</i>	select ETI2 as clock source
<i>GPTIMER_CLOCK_SOURCE_ETI3</i>	select ETI3 as clock source
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select a GPTIMER0 clock source */
```

```
gptimer_clock_source_select(GPTIMER0, GPTIMER_CLOCK_SOURCE_CK_GPTIMER);
```

### **`gptimer_clock_polarity_config`**

The description of `gptimer_clock_polarity_config` is shown as below:

Table 3-469. Function `gptimer_clock_polarity_config`

<b>Function name</b>	<code>gptimer_clock_polarity_config</code>
<b>Function prototype</b>	<code>void gptimer_clock_polarity_config(uint32_t timer_periph, uint32_t clock_polarity)</code>
<b>Function descriptions</b>	configure a GPTIMER clock polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral

<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>clock_polarity</b>	clock polarity select
<i>GPTIMER_CLOCK_POLARITY_RISING</i>	select rising edge as clock polarity
<i>GPTIMER_CLOCK_POLARITY_FALLING</i>	select falling edge as clock polarity
<i>GPTIMER_CLOCK_POLARITY_BOTH</i>	select both edge as clock polarity
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure a GPTIMER0 clock polarity */
```

```
gptimer_clock_polarity_config(GPTIMER0, GPTIMER_CLOCK_POLARITY_RISING);
```

### **gptimer\_counter\_software\_enable**

The description of gptimer\_counter\_software\_enable is shown as below:

**Table 3-470. Function gptimer\_counter\_software\_enable**

<b>Function name</b>	gptimer_counter_software_enable
<b>Function prototype</b>	void gptimer_counter_software_enable(uint32_t timer_periph, uint32_t timer_cnt)
<b>Function descriptions</b>	software enable GPTIMER counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>timer_cnt</b>	GPTIMER counter
<i>GPTIMER0_COUNT</i>	GPTIMER0 counter
<i>GPTIMER1_COUNT</i>	GPTIMER1 counter
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* software enable GPTIMER0 counter */
```



```
gptimer_counter_software_enable(GPTIMER0, GPTIMER0_COUNT);
```

### gptimer\_counter\_software\_disable

The description of gptimer\_counter\_software\_disable is shown as below:

**Table 3-471. Function gptimer\_counter\_software\_disable**

<b>Function name</b>	gptimer_counter_software_disable
<b>Function prototype</b>	void gptimer_counter_software_disable(uint32_t timer_periph, uint32_t timer_cnt)
<b>Function descriptions</b>	software disable GPTIMER counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>timer_cnt</b>	GPTIMER counter
<i>GPTIMER0_COUNT</i>	GPTIMER0 counter
<i>GPTIMER1_COUNT</i>	GPTIMER1 counter
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* software disable GPTIMER0 counter */
```

```
gptimer_counter_software_disable(GPTIMER0, GPTIMER0_COUNT);
```

### gptimer\_counter\_software\_reset

The description of gptimer\_counter\_software\_reset is shown as below:

**Table 3-472. Function gptimer\_counter\_software\_reset**

<b>Function name</b>	gptimer_counter_software_reset
<b>Function prototype</b>	void gptimer_counter_software_reset(uint32_t timer_periph, uint32_t timer_cnt)
<b>Function descriptions</b>	software reset GPTIMER counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>timer_cnt</b>	GPTIMER counter

<i>GPTIMER0_COUNT</i>	GPTIMER0 counter
<i>GPTIMER1_COUNT</i>	GPTIMER1 counter
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* software reset GPTIMER0 counter */
```

```
gptimer_counter_software_reset(GPTIMER0, GPTIMER0_COUNT);
```

### **gptimer\_counter\_enable\_source\_struct\_para\_init**

The description of `gptimer_counter_enable_source_struct_para_init` is shown as below:

**Table 3-473. Function `gptimer_counter_enable_source_struct_para_init`**

<b>Function name</b>	<code>gptimer_counter_enable_source_struct_para_init</code>
<b>Function prototype</b>	void <code>gptimer_counter_enable_source_struct_para_init(gptimer_counter_enable_source_parameter_struct *counter_enable_source_para)</code>
<b>Function descriptions</b>	initialize GPTIMER counter enable source init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>counter_enable_source_para</b>	counter enable source init parameter struct, refer to <a href="#">Table 3-455. Structure <code>gptimer_counter_enable_source_parameter_struct</code></a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* init parameter struct with a default value */
```

```
gptimer_counter_enable_source_parameter_struct para;
```

```
gptimer_counter_enable_source_struct_para_init(&para);
```

### **gptimer\_counter\_enable\_source\_config**

The description of `gptimer_counter_enable_source_config` is shown as below:

**Table 3-474. Function `gptimer_counter_enable_source_config`**

<b>Function name</b>	<code>gptimer_counter_enable_source_config</code>
<b>Function prototype</b>	void <code>gptimer_counter_enable_source_config(uint32_t timer_periph,</code>

	gptimer_counter_enable_source_parameter_struct *counter_enable_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter enable source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<b>GPTIMERx</b>	x=0,1
<b>Input parameter{in}</b>	
<b>counter_enable_source_e_para</b>	counter enable source init parameter struct, refer to <a href="#">Table 3-455. Structure gptimer counter enable source parameter struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize GPTIMER0 counter enable source */
```

```
gptimer_counter_enable_source_parameter_struct para;
```

```
gptimer_counter_enable_source_config(GPTIMER0, &para);
```

### gptimer\_counter\_disable\_source\_struct\_para\_init

The description of gptimer\_counter\_disable\_source\_struct\_para\_init is shown as below:

**Table 3-475. Function gptimer\_counter\_disable\_source\_struct\_para\_init**

<b>Function name</b>	gptimer_counter_disable_source_struct_para_init
<b>Function prototype</b>	void gptimer_counter_disable_source_struct_para_init(gptimer_counter_disable_source_parameter_struct *counter_disable_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter disable source init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>counter_disable_source_e_para</b>	counter disable source init parameter struct, refer to <a href="#">Table 3-456. Structure gptimer counter disable source parameter struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* init parameter struct with a default value */
```

```
gptimer_counter_disable_source_parameter_struct para;
```

```
gptimer_counter_disable_source_struct_para_init(&para);
```

### gptimer\_counter\_disable\_source\_config

The description of gptimer\_counter\_disable\_source\_config is shown as below:

**Table 3-476. Function gptimer\_counter\_disable\_source\_config**

<b>Function name</b>	gptimer_counter_disable_source_config
<b>Function prototype</b>	void gptimer_counter_disable_source_config(uint32_t timer_periph, gptimer_counter_disable_source_parameter_struct *counter_disable_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter disable source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>counter_disable_source_para</b>	counter disable source init parameter struct, refer to <a href="#">Table 3-456. Structure gptimer counter disable source parameter struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize GPTIMER0 counter disable source */
```

```
gptimer_counter_disable_source_parameter_struct para;
```

```
gptimer_counter_disable_source_config(GPTIMER0, &para);
```

### gptimer\_counter\_reset\_source\_struct\_para\_init

The description of gptimer\_counter\_reset\_source\_struct\_para\_init is shown as below:

**Table 3-477. Function gptimer\_counter\_reset\_source\_struct\_para\_init**

<b>Function name</b>	gptimer_counter_reset_source_struct_para_init
<b>Function prototype</b>	void gptimer_counter_reset_source_struct_para_init(gptimer_counter_reset_source_parameter_struct *counter_reset_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter reset source init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>counter_reset_source _para</b>	counter reset source init parameter struct, refer to <a href="#">Table 3-457. Structure gptimer counter reset source parameter struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* init parameter struct with a default value */
gptimer_counter_reset_source_parameter_struct para;
gptimer_counter_reset_source_struct_para_init(&para);
```

### gptimer\_counter\_reset\_source\_config

The description of gptimer\_counter\_reset\_source\_config is shown as below:

**Table 3-478. Function gptimer\_counter\_reset\_source\_config**

<b>Function name</b>	gptimer_counter_reset_source_config
<b>Function prototype</b>	void gptimer_counter_reset_source_config(uint32_t timer_periph, gptimer_counter_reset_source_parameter_struct *counter_reset_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter reset source
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
Input parameter{in}	
<b>counter_reset_source _para</b>	counter reset source init parameter struct, refer to <a href="#">Table 3-457. Structure gptimer counter reset source parameter struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize GPTIMER0 counter reset source */
gptimer_counter_rest_source_parameter_struct para;
gptimer_counter_reset_source_config(GPTIMER0, &para);
```

## gptimer\_counter\_up\_source\_struct\_para\_init

The description of gptimer\_counter\_up\_source\_struct\_para\_init is shown as below:

**Table 3-479. Function gptimer\_counter\_up\_source\_struct\_para\_init**

<b>Function name</b>	gptimer_counter_up_source_struct_para_init
<b>Function prototype</b>	void gptimer_counter_up_source_struct_para_init(gptimer_counter_up_source_parameter_struct *counter_up_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter up source init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>counter_up_source_para</b>	counter up source init parameter struct, refer to <a href="#">Table 3-459. Structure gptimer_counter_up_source_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* init parameter struct with a default value */
gptimer_counter_up_source_parameter_struct para;
gptimer_counter_up_source_struct_para_init(&para);
```

## gptimer\_counter\_up\_source\_config

The description of gptimer\_counter\_up\_source\_config is shown as below:

**Table 3-480. Function gptimer\_counter\_up\_source\_config**

<b>Function name</b>	gptimer_counter_up_source_config
<b>Function prototype</b>	void gptimer_counter_up_source_config(uint32_t timer_periph, gptimer_counter_up_source_parameter_struct *counter_up_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter up source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<b>GPTIMERx</b>	x=0,1
<b>Input parameter{in}</b>	
<b>counter_up_source_para</b>	counter up source init parameter struct, refer to <a href="#">Table 3-459. Structure gptimer_counter_up_source_parameter_struct</a>
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* initialize GPTIMER0 counter up source */
```

```
gptimer_counter_up_source_parameter_struct para;
```

```
gptimer_counter_up_source_config(GPTIMER0, &para);
```

### **gptimer\_counter\_down\_source\_struct\_para\_init**

The description of gptimer\_counter\_down\_source\_struct\_para\_init is shown as below:

**Table 3-481. Function gptimer\_counter\_down\_source\_struct\_para\_init**

<b>Function name</b>	gptimer_counter_down_source_struct_para_init
<b>Function prototype</b>	void gptimer_counter_down_source_struct_para_init(gptimer_counter_down_source_parameter_struct *counter_down_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter down source init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>counter_down_source_para</b>	counter down source init parameter struct, refer to <a href="#">Table 3-460. Structure gptimer_counter_down_source_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* init parameter struct with a default value */
```

```
gptimer_counter_down_source_parameter_struct para;
```

```
gptimer_counter_down_source_struct_para_init(&para);
```

### **gptimer\_counter\_down\_source\_config**

The description of gptimer\_counter\_down\_source\_config is shown as below:

**Table 3-482. Function gptimer\_counter\_down\_source\_config**

<b>Function name</b>	gptimer_counter_down_source_config
<b>Function prototype</b>	void gptimer_counter_down_source_config(uint32_t timer_periph, gptimer_counter_down_source_parameter_struct *counter_down_source_para)

<b>Function descriptions</b>	initialize GPTIMER counter down source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>counter_down_source _para</b>	counter down source init parameter struct, refer to <a href="#">Table 3-460. Structure gptimer_counter_down_source_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize GPTIMER0 counter down source */
```

```
gptimer_counter_down_source_parameter_struct para;
```

```
gptimer_counter_down_source_config(GPTIMER0, &para);
```

### **gptimer\_counter\_up\_input\_level\_source\_select**

The description of gptimer\_counter\_up\_input\_level\_source\_select is shown as below:

**Table 3-483. Function gptimer\_counter\_up\_input\_level\_source\_select**

<b>Function name</b>	gptimer_counter_up_input_level_source_select
<b>Function prototype</b>	void gptimer_counter_up_input_level_source_select(uint32_t timer_periph, uint32_t input_source)
<b>Function descriptions</b>	select counter up input level source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>input_source</b>	counter up input level source
<i>GPTIMER_COUNTER_UP_DISABLE</i>	counter up input source disable
<i>GPTIMER_COUNTER_UP_CH0_LOW</i>	channel 0 low as counter up input source enable
<i>GPTIMER_COUNTER_UP_CH0_HIGH</i>	channel 0 high as counter up input source enable
<i>GPTIMER_COUNTER_UP_CH1_LOW</i>	channel 1 low as counter up input source enable



<code>GPTIMER_COUNTER_UP_CH1_HIGH</code>	channel 1 high as counter up input source enable
<code>GPTIMER_COUNTER_UP_ETI0_LOW</code>	eti0 low as counter up input source enable
<code>GPTIMER_COUNTER_UP_ETI0_HIGH</code>	eti0 high as counter up input source enable
<code>GPTIMER_COUNTER_UP_ETI1_LOW</code>	eti1 low as counter up input source enable
<code>GPTIMER_COUNTER_UP_ETI1_HIGH</code>	eti1 high as counter up input source enable
<code>GPTIMER_COUNTER_UP_ETI2_LOW</code>	eti2 low as counter up input source enable
<code>GPTIMER_COUNTER_UP_ETI2_HIGH</code>	eti2 high as counter up input source enable
<code>GPTIMER_COUNTER_UP_ETI3_LOW</code>	eti3 low as counter up input source enable
<code>GPTIMER_COUNTER_UP_ETI3_HIGH</code>	eti3 high as counter up input source enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select counter up input level source */
```

```
gptimer_counter_up_input_level_source_select(GPTIMER0,
GPTIMER_COUNTER_UP_CH0_HIGH);
```

### **gptimer\_counter\_down\_input\_level\_source\_select**

The description of `gptimer_counter_down_input_level_source_select` is shown as below:

**Table 3-484. Function `gptimer_counter_down_input_level_source_select`**

<b>Function name</b>	<code>gptimer_counter_down_input_level_source_select</code>
<b>Function prototype</b>	<code>void gptimer_counter_down_input_level_source_select(uint32_t timer_periph, uint32_t input_source)</code>
<b>Function descriptions</b>	select counter down input level source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<b>GPTIMERx</b>	x=0,1
<b>Input parameter{in}</b>	
<b>input_source</b>	counter down input level source

<code>GPTIMER_COUNTER_DOWN_DISABLE</code>	counter down input source disable
<code>GPTIMER_COUNTER_DOWN_CH0_LOW</code>	channel 0 low as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_CH0_HIGH</code>	channel 0 high as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_CH1_LOW</code>	channel 1 low as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_CH1_HIGH</code>	channel 1 high as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_ETI0_LOW</code>	eti0 low as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_ETI0_HIGH</code>	eti0 high as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_ETI1_LOW</code>	eti1 low as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_ETI1_HIGH</code>	eti1 high as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_ETI2_LOW</code>	eti2 low as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_ETI2_HIGH</code>	eti2 high as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_ETI3_LOW</code>	eti3 low as counter down input source enable
<code>GPTIMER_COUNTER_DOWN_ETI3_HIGH</code>	eti3 high as counter down input source enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select counter down input level source */
```

```
gptimer_counter_down_input_level_source_select(GPTIMER0,
GPTIMER_COUNTER_DOWN_CH0_LOW);
```

### **gptimer\_enable**

The description of `gptimer_enable` is shown as below:

**Table 3-485. Function `gptimer_enable`**

<b>Function name</b>	<code>gptimer_enable</code>
<b>Function prototype</b>	<code>void gptimer_enable(uint32_t timer_periph)</code>
<b>Function descriptions</b>	enable GPTIMER

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 */
gptimer_enable(GPTIMER0);
```

### **gptimer\_disable**

The description of gptimer\_disable is shown as below:

**Table 3-486. Function gptimer\_disable**

<b>Function name</b>	gptimer_disable
<b>Function prototype</b>	void gptimer_disable(uint32_t timer_periph)
<b>Function descriptions</b>	disable GPTIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 */
gptimer_disable(GPTIMER0);
```

### **gptimer\_auto\_reload\_shadow\_enable**

The description of gptimer\_auto\_reload\_shadow\_enable is shown as below:

**Table 3-487. Function gptimer\_auto\_reload\_shadow\_enable**

<b>Function name</b>	gptimer_auto_reload_shadow_enable
<b>Function prototype</b>	void gptimer_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>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 the auto reload shadow function */
```

```
gptimer_auto_reload_shadow_enable(GPTIMER0);
```

### **gptimer\_auto\_reload\_shadow\_disable**

The description of gptimer\_auto\_reload\_shadow\_disable is shown as below:

**Table 3-488. Function gptimer\_auto\_reload\_shadow\_disable**

<b>Function name</b>	gptimer_auto_reload_shadow_disable
<b>Function prototype</b>	void gptimer_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>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 the auto reload shadow function */
```

```
gptimer_auto_reload_shadow_disable(GPTIMER0);
```

### **gptimer\_update\_event\_enable**

The description of gptimer\_update\_event\_enable is shown as below:

**Table 3-489. Function gptimer\_update\_event\_enable**

<b>Function name</b>	gptimer_update_event_enable
<b>Function prototype</b>	void gptimer_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>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 the update event */
```

```
gptimer_update_event_enable(GPTIMER0);
```

### **gptimer\_update\_event\_disable**

The description of gptimer\_update\_event\_disable is shown as below:

**Table 3-490. Function gptimer\_update\_event\_disable**

<b>Function name</b>	gptimer_update_event_disable
<b>Function prototype</b>	void gptimer_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>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 the update event */
```

```
gptimer_update_event_disable(GPTIMER0);
```

### **gptimer\_counter\_alignment**

The description of gptimer\_counter\_alignment is shown as below:

**Table 3-491. Function gptimer\_counter\_alignment**

<b>Function name</b>	gptimer_counter_alignment
<b>Function prototype</b>	void gptimer_counter_alignment(uint32_t timer_periph, uint16_t aligned)
<b>Function descriptions</b>	set counter alignment mode

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>aligned</b>	counter alignment mode
<i>GPTIMER_COUNTER_EDGE</i>	edge-aligned mode
<i>GPTIMER_COUNTER_CENTER</i>	center-aligned mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set GPTIMER0 counter alignment mode */
```

```
gptimer_counter_alignment(GPTIMER0);
```

### **gptimer\_counter\_up\_direction**

The description of gptimer\_counter\_up\_direction is shown as below:

**Table 3-492. Function gptimer\_counter\_up\_direction**

<b>Function name</b>	gptimer_counter_up_direction
<b>Function prototype</b>	void gptimer_counter_up_direction(uint32_t timer_periph)
<b>Function descriptions</b>	set counter up direction
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set GPTIMER0 counter up direction */
```

```
gptimer_counter_up_direction(GPTIMER0);
```

## gptimer\_counter\_down\_direction

The description of gptimer\_counter\_down\_direction is shown as below:

**Table 3-493. Function gptimer\_counter\_down\_direction**

<b>Function name</b>	gptimer_counter_down_direction
<b>Function prototype</b>	void gptimer_counter_down_direction(uint32_t timer_periph)
<b>Function descriptions</b>	set counter down direction
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set GPTIMER0 counter down direction */
gptimer_counter_down_direction(GPTIMER0);
```

## gptimer\_counter\_direction\_force\_set\_enable

The description of gptimer\_counter\_direction\_force\_set\_enable is shown as below:

**Table 3-494. Function gptimer\_counter\_direction\_force\_set\_enable**

<b>Function name</b>	gptimer_counter_direction_force_set_enable
<b>Function prototype</b>	void gptimer_counter_direction_force_set_enable(uint32_t timer_periph)
<b>Function descriptions</b>	enable counter direction force set
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 counter direction force set */
gptimer_counter_direction_force_set_enable(GPTIMER0);
```

## gptimer\_counter\_direction\_force\_set\_disable

The description of gptimer\_counter\_direction\_force\_set\_disable is shown as below:

**Table 3-495. Function gptimer\_counter\_direction\_force\_set\_disable**

<b>Function name</b>	gptimer_counter_direction_force_set_disable
<b>Function prototype</b>	void gptimer_counter_direction_force_set_disable(uint32_t timer_periph)
<b>Function descriptions</b>	disable counter direction force set
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 counter direction force set */
```

```
gptimer_counter_direction_force_set_disable(GPTIMER0);
```

## gptimer\_register\_global\_update\_source\_select

The description of gptimer\_register\_global\_update\_source\_select is shown as below:

**Table 3-496. Function gptimer\_register\_global\_update\_source\_select**

<b>Function name</b>	gptimer_register_global_update_source_select
<b>Function prototype</b>	void gptimer_register_global_update_source_select(uint32_t timer_periph, uint32_t source)
<b>Function descriptions</b>	select register global update source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>source</b>	global update source
<i>GPTIMER_GLOBAL_UPS_SEL_FLOW</i>	global update event source select counter overflow / underflow event
<i>GPTIMER_GLOBAL_UPS_SEL_OVERFLOW</i>	global update event source select counter overflow event
<i>GPTIMER_GLOBAL_UPS_SEL_UNDERFLOW</i>	global update event source select counter underflow event
<b>Output parameter{out}</b>	



-	-
Return value	
-	-

Example:

```
/* select register global update source */
```

```
gptimer_register_global_update_source_select(GPTIMER0,
GPTIMER_GLOBAL_UPS_SEL_FLOW);
```

### **gptimer\_register\_local\_update\_source\_select**

The description of gptimer\_register\_local\_update\_source\_select is shown as below:

**Table 3-497. Function gptimer\_register\_local\_update\_source\_select**

<b>Function name</b>	gptimer_register_local_update_source_select
<b>Function prototype</b>	void gptimer_register_local_update_source_select(uint32_t timer_periph, uint32_t reg, uint32_t source)
<b>Function descriptions</b>	select register local update source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>reg</b>	local update source
<i>GPTIMER_LOCAL_UP_CH0CV</i>	CH0CV local update
<i>GPTIMER_LOCAL_UP_CH0COMV_ADD</i>	CH0COMV_ADD local update
<i>GPTIMER_LOCAL_UP_CH1CV</i>	CH1CV local update
<i>GPTIMER_LOCAL_UP_CH1COMV_ADD</i>	CH1COMV_ADD local update
<i>GPTIMER_LOCAL_UP_ADCCR1</i>	ADC triggered comparison value 1 local update
<i>GPTIMER_LOCAL_UP_ADCCR2</i>	ADC triggered comparison value 2 local update
<i>GPTIMER_LOCAL_UP_CAR</i>	car local update
<i>GPTIMER_LOCAL_UP_CH0</i>	channel 0 output mode local update
<i>GPTIMER_LOCAL_UP_CH1</i>	channel 1 output mode local update
<b>Input parameter{in}</b>	

source	update source
<i>GPTIMER_LOCAL_UP_S_SEL_DISABLE</i>	local update event source select disable
<i>GPTIMER_LOCAL_UP_S_SEL_OVERFLOW</i>	local update event source select counter overflow event
<i>GPTIMER_LOCAL_UP_S_SEL_UNDERFLOW</i>	local update event source select counter underflow event
<i>GPTIMER_LOCAL_UP_S_SEL_FLOW</i>	local update event source select counter flow event
<i>GPTIMER_LOCAL_UP_S_SEL_CHCOM</i>	local update event source select channel compare event(only GPTIMER_LOCAL_UP_CH0COMV_ADD)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select register local update source */
```

```
gptimer_register_local_update_source_select(GPTIMER0, GPTIMER_LOCAL_UP_CH0CV,  
GPTIMER_LOCAL_UPS_SEL_OVERFLOW);
```

### **gptimer\_prescaler\_config**

The description of gptimer\_prescaler\_config is shown as below:

**Table 3-498. Function gptimer\_prescaler\_config**

<b>Function name</b>	gptimer_prescaler_config
<b>Function prototype</b>	void gptimer_prescaler_config(uint32_t timer_periph, uint16_t prescaler)
<b>Function descriptions</b>	configure prescaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>prescaler</b>	prescaler value,0~65535
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 prescaler */
```

```
gptimer_prescaler_config(GPTIMER0, 0x3fff);
```

## gptimer\_update\_repetition\_value\_config

The description of gptimer\_update\_repetition\_value\_config is shown as below:

**Table 3-499. Function gptimer\_update\_repetition\_value\_config**

<b>Function name</b>	gptimer_update_repetition_value_config
<b>Function prototype</b>	void gptimer_update_repetition_value_config(uint32_t timer_periph, uint16_t repetition, uint32_t load_mode)
<b>Function descriptions</b>	configure update repetition register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>repetition</b>	the counter repetition value, 0~4095
<b>Input parameter{in}</b>	
<b>load_mode</b>	loading mode
<i>GPTIMER_UPREP_LO AD_IMMEDIATELY</i>	repetition load immediately
<i>GPTIMER_UPREP_LO AD_NEXT_UPEVENT</i>	repetition load at next update
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 repetition register value */
```

```
gptimer_update_repetition_value_config(GPTIMER0,                                0,  
GPTIMER_UPREP_LOAD_IMMEDIATELY);
```

## gptimer\_update\_repetition\_counter\_read

The description of gptimer\_update\_repetition\_counter\_read is shown as below:

**Table 3-500. Function gptimer\_update\_repetition\_counter\_read**

<b>Function name</b>	gptimer_update_repetition_counter_read
<b>Function prototype</b>	uint16_t gptimer_update_repetition_counter_read(uint32_t timer_periph)
<b>Function descriptions</b>	read update repetition register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1

Output parameter{out}	
-	-
Return value	
uint16_t	repetition value

Example:

```
/* read GPTIMER0 repetition register value */
```

```
uint16_t value;
```

```
value = gptimer_update_repetition_counter_read (GPTIMER0);
```

### **gptimer\_autoreload\_value\_config**

The description of gptimer\_autoreload\_value\_config is shown as below:

**Table 3-501. Function gptimer\_autoreload\_value\_config**

Function name	gptimer_autoreload_value_config
Function prototype	void gptimer_autoreload_value_config(uint32_t timer_periph, uint32_t autoreload)
Function descriptions	configure autoreload register value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	GPTIMER peripheral
GPTIMERx	x=0,1
Input parameter{in}	
autoreload	the counter auto-reload value,0~65535
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPTIMER0 autoreload register value */
```

```
gptimer_autoreload_value_config (GPTIMER0, 0xff);
```

### **gptimer\_counter\_value\_config**

The description of gptimer\_counter\_value\_config is shown as below:

**Table 3-502. Function gptimer\_counter\_value\_config**

Function name	gptimer_counter_value_config
Function prototype	void gptimer_counter_value_config(uint32_t timer_periph, uint32_t counter)
Function descriptions	configure counter register value

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>counter</b>	the counter value, 0~65535
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 counter register value */
```

```
gptimer_counter_value_config (GPTIMER0, 0xff);
```

### **gptimer\_counter\_read**

The description of gptimer\_counter\_read is shown as below:

**Table 3-503. Function gptimer\_counter\_read**

<b>Function name</b>	gptimer_counter_read
<b>Function prototype</b>	uint32_t gptimer_counter_read(uint32_t timer_periph)
<b>Function descriptions</b>	read counter value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	counter value

Example:

```
/* read GPTIMER0 counter value */
```

```
uint32_t value;
```

```
value = gptimer_counter_read(GPTIMER0);
```

### **gptimer\_prescaler\_read**

The description of gptimer\_prescaler\_read is shown as below:

Table 3-504. Function `gptimer_prescaler_read`

Function name	<code>gptimer_prescaler_read</code>
Function prototype	<code>uint16_t gptimer_prescaler_read(uint32_t timer_periph)</code>
Function descriptions	read prescaler value
Precondition	-
The called functions	-
Input parameter{in}	
<code>timer_periph</code>	GPTIMER peripheral
<code>GPTIMERx</code>	<code>x=0,1</code>
Output parameter{out}	
-	-
Return value	
<code>uint16_t</code>	prescaler register value

Example:

```
/* read GPTIMER0 prescaler value */
```

```
uint16_t value;
```

```
value = gptimer_prescaler_read (GPTIMER0);
```

### `gptimer_single_pulse_mode_config`

The description of `gptimer_single_pulse_mode_config` is shown as below:

Table 3-505. Function `gptimer_single_pulse_mode_config`

Function name	<code>gptimer_single_pulse_mode_config</code>
Function prototype	<code>void gptimer_single_pulse_mode_config(uint32_t timer_periph, uint32_t spmode)</code>
Function descriptions	configure single pulse mode
Precondition	-
The called functions	-
Input parameter{in}	
<code>timer_periph</code>	GPTIMER peripheral
<code>GPTIMERx</code>	<code>x=0,1</code>
Input parameter{in}	
<code>spmode</code>	single pulse mode select
<code>GPTIMER_SP_MODE_SINGLE</code>	single pulse mode
<code>GPTIMER_SP_MODE_REPETITIVE</code>	repetitive pulse mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPTIMER0 single pulse mode */
```

```
gptimer_single_pulse_mode_config (GPTIMER0, GPTIMER_SP_MODE_SINGLE);
```

### **gptimer\_dma\_enable**

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

**Table 3-506. Function gptimer\_dma\_enable**

<b>Function name</b>	gptimer_dma_enable
<b>Function prototype</b>	void gptimer_dma_enable(uint32_t timer_periph, uint32_t dma)
<b>Function descriptions</b>	enable DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>dma</b>	DMA request
<i>GPTIMER_DMA_CH0D</i>	channel 0 capture or compare DMA request
<i>GPTIMER_DMA_CH1D</i>	channel 1 capture or compare DMA request
<i>GPTIMER_DMA_CH0C</i> <i>OMADD</i>	channel 0 additional compare DMA request
<i>GPTIMER_DMA_CH1C</i> <i>OMADD</i>	channel 1 additional compare DMA request
<i>GPTIMER_DMA_REPE</i> <i>D</i>	repetition count end DMA request
<i>GPTIMER_DMA_OVER</i> <i>FLOWD</i>	counter overflow DMA request
<i>GPTIMER_DMA_UNDE</i> <i>RFLOWD</i>	counter underflow DMA request
<i>GPTIMER_DMA_OVUN</i> <i>FLOWD</i>	counter overflow/underflow DMA request
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DMA */
```

```
gptimer_dma_enable (GPTIMER0, GPTIMER_DMA_CH0D);
```

## gptimer\_dma\_disable

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

**Table 3-507. Function gptimer\_dma\_disable**

<b>Function name</b>	gptimer_dma_disable
<b>Function prototype</b>	void gptimer_dma_disable(uint32_t timer_periph, uint32_t dma)
<b>Function descriptions</b>	disable DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>dma</b>	DMA request
<i>GPTIMER_DMA_CH0D</i>	channel 0 capture or compare DMA request
<i>GPTIMER_DMA_CH1D</i>	channel 1 capture or compare DMA request
<i>GPTIMER_DMA_CH0C</i> <i>OMADD</i>	channel 0 additional compare DMA request
<i>GPTIMER_DMA_CH1C</i> <i>OMADD</i>	channel 1 additional compare DMA request
<i>GPTIMER_DMA_REPE</i> <i>D</i>	repetition count end DMA request
<i>GPTIMER_DMA_OVERFLOW</i> <i>FLOWD</i>	counter overflow DMA request
<i>GPTIMER_DMA_UNDER</i> <i>RFLOWD</i>	counter underflow DMA request
<i>GPTIMER_DMA_OVUN</i> <i>FLOWD</i>	counter overflow/underflow DMA request
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMA */
```

```
gptimer_dma_disable (GPTIMER0, GPTIMER_DMA_CH0D);
```

## gptimer\_dma\_transfer\_config

The description of gptimer\_dma\_transfer\_config is shown as below:

**Table 3-508. Function gptimer\_dma\_transfer\_config**

<b>Function name</b>	gptimer_dma_transfer_config
<b>Function prototype</b>	void gptimer_dma_transfer_config(uint32_t timer_periph, uint32_t



	dma_baseaddr, uint32_t dma_lenth)
<b>Function descriptions</b>	configure the DMA transfer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>dma_baseaddr</b>	DMA transfer address
<i>GPTIMER_DMACFG_D MATA_WP</i>	DMA transfer address is GPTIMER_WP
<i>GPTIMER_DMACFG_D MATA_SWEN</i>	DMA transfer address is GPTIMER_SWEN
<i>GPTIMER_DMACFG_D MATA_SWDIS</i>	DMA transfer address is GPTIMER_SWDIS
<i>GPTIMER_DMACFG_D MATA_SWRST</i>	DMA transfer address is GPTIMER_SWRST
<i>GPTIMER_DMACFG_D MATA_ESSEL</i>	DMA transfer address is GPTIMER_ESSEL
<i>GPTIMER_DMACFG_D MATA_DSSEL</i>	DMA transfer address is GPTIMER_DSSEL
<i>GPTIMER_DMACFG_D MATA_RSSEL</i>	DMA transfer address is GPTIMER_RSSEL
<i>GPTIMER_DMACFG_D MATA_CH0CSSEL</i>	DMA transfer address is GPTIMER_CH0CSSEL
<i>GPTIMER_DMACFG_D MATA_CH1CSSEL</i>	DMA transfer address is GPTIMER_CH0CSSEL
<i>GPTIMER_DMACFG_D MATA_CTL1</i>	DMA transfer address is GPTIMER_CTL0
<i>GPTIMER_DMACFG_D MATA_UPSSR</i>	DMA transfer address is GPTIMER_CUPEVSSEL
<i>GPTIMER_DMACFG_D MATA_DNSSR</i>	DMA transfer address is GPTIMER_CDNEVSSEL
<i>GPTIMER_DMACFG_D MATA_CH0CTL</i>	DMA transfer address is GPTIMER_CH0CTL
<i>GPTIMER_DMACFG_D MATA_CH1CTL</i>	DMA transfer address is GPTIMER_CH1CTL
<i>GPTIMER_DMACFG_D MATA_CHCTL</i>	DMA transfer address is GPTIMER_CHCTL
<i>GPTIMER_DMACFG_D MATA_DMAINTEN</i>	DMA transfer address is GPTIMER_DMAINTEN
<i>GPTIMER_DMACFG_D MATA_INTF</i>	DMA transfer address is GPTIMER_INTF

GPTIMER_DMACFG_D MATA_UPSSEL	DMA transfer address is GPTIMER_UPSSEL
GPTIMER_DMACFG_D MATA_CNT	DMA transfer address is GPTIMER_CNT
GPTIMER_DMACFG_D MATA_PSC	DMA transfer address is GPTIMER_PSC
GPTIMER_DMACFG_D MATA_CAR	DMA transfer address is GPTIMER_CAR
GPTIMER_DMACFG_D MATA_CH0CV	DMA transfer address is GPTIMER_CH0CV
GPTIMER_DMACFG_D MATA_CH1CV	DMA transfer address is GPTIMER_CH1CV
GPTIMER_DMACFG_D MATA_CH0COMV_AD D	DMA transfer address is GPTIMER_CH0COMV_ADD
GPTIMER_DMACFG_D MATA_CH1COMV_AD D	DMA transfer address is GPTIMER_CH1COMV_ADD
GPTIMER_DMACFG_D MATA_DTCTL	DMA transfer address is GPTIMER_DTCTL
GPTIMER_DMACFG_D MATA_ADCTL	DMA transfer address is GPTIMER_ADCTL
GPTIMER_DMACFG_D MATA_ADCCR1	DMA transfer address is GPTIMER_ADCCR1
GPTIMER_DMACFG_D MATA_ADCCR2	DMA transfer address is GPTIMER_ADCCR2
GPTIMER_DMACFG_D MATA_ADCTRGS	DMA transfer address is GPTIMER_ADCTRGS
GPTIMER_DMACFG_D MATA_ADDINTSCTL0	DMA transfer address is GPTIMER_ADDINTSCTL0
GPTIMER_DMACFG_D MATA_ADDINTSCTL1	DMA transfer address is GPTIMER_ADDINTSCTL1
GPTIMER_DMACFG_D MATA_IADCTSS	DMA transfer address is GPTIMER_IADCTSS
GPTIMER_DMACFG_D MATA_CREP	DMA transfer address is GPTIMER_CREP
GPTIMER_DMACFG_D MATA_SYNRSTCTL	DMA transfer address is GPTIMER_SYNRSTCTL
<b>Input parameter{in}</b>	
<b>dma_lenth</b>	DMA transfer lenth
GPTIMER_DMACFG_D MATC_xTRANSFER(x= 1..35)	DMA transfer x time

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the GPTIMER0 DMA transfer */
```

```
gptimer_dma_transfer_config (GPTIMER0, GPTIMER_DMACFG_DMATA_ADDINTSCTL0,
GPTIMER_DMACFG_DMATC_2TRANSFER);
```

### **gptimer\_channel\_output\_struct\_para\_init**

The description of gptimer\_channel\_output\_struct\_para\_init is shown as below:

**Table 3-509. Function gptimer\_channel\_output\_struct\_para\_init**

Function name	gptimer_channel_output_struct_para_init
Function prototype	void gptimer_channel_output_struct_para_init(gptimer_oc_parameter_struct *ocpara)
Function descriptions	initialize GPTIMER channel output parameter struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
ocpara	GPTIMER channel n output parameter struct, refer to <a href="#">Table 3-461. Structure gptimer oc parameter struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* init parameter struct with a default value */
```

```
gptimer_oc_parameter_struct para;
```

```
gptimer_channel_output_struct_para_init (&para);
```

### **gptimer\_channel\_output\_config**

The description of gptimer\_channel\_output\_config is shown as below:

**Table 3-510. Function gptimer\_channel\_output\_config**

Function name	gptimer_channel_output_config
Function prototype	void gptimer_channel_output_config(uint32_t timer_periph, uint16_t channel, gptimer_oc_parameter_struct *ocpara)
Function descriptions	configure channel output mode

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
<b>Input parameter{in}</b>	
<b>ocpara</b>	GPTIMER channel n output parameter struct, refer to <a href="#">Table 3-461. Structure gptimer_oc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 channel output mode */
```

```
gptimer_oc_parameter_struct para;
```

```
gptimer_channel_output_config (GPTIMER0, GPTIMER_CH0, &para);
```

### **gptimer\_channel\_output\_force\_duty\_config**

The description of gptimer\_channel\_output\_force\_duty\_config is shown as below:

**Table 3-511. Function gptimer\_channel\_output\_force\_duty\_config**

<b>Function name</b>	gptimer_channel_output_force_duty_config
<b>Function prototype</b>	void gptimer_channel_output_force_duty_config(uint32_t timer_periph, uint16_t channel, uint32_t duty, uint32_t output_time)
<b>Function descriptions</b>	configure channel output force duty
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
<b>Input parameter{in}</b>	
<b>duty</b>	duty mode
<i>GPTIMER_COMPARE_</i>	output duty by compare

<i>DUTY_OUTPUT</i>	
<i>GPTIMER_FORCE_0_DUTY_OUTPUT</i>	output duty by force 0%
<i>GPTIMER_FORCE_100_DUTY_OUTPUT</i>	output duty by force 100%
<b>Input parameter{in}</b>	
<b>output_time</b>	output time
<i>GPTIMER_FORCE_DUTY_OUTPUT_IMMEDIATELY</i>	force duty output immediately
<i>GPTIMER_FORCE_DUTY_OUTPUT_NEXT_PERIOD</i>	force duty output at next period
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 channel output force duty */
```

```
gptimer_channel_output_force_duty_config(GPTIMER0, GPTIMER_CH0,
GPTIMER_COMPARE_DUTY_OUTPUT,
GPTIMER_FORCE_DUTY_OUTPUT_IMMEDIATELY);
```

### **gptimer\_channel\_output\_compare\_value\_config**

The description of `gptimer_channel_output_compare_value_config` is shown as below:

**Table 3-512. Function `gptimer_channel_output_compare_value_config`**

<b>Function name</b>	<code>gptimer_channel_output_compare_value_config</code>
<b>Function prototype</b>	<code>void gptimer_channel_output_compare_value_config(uint32_t timer_periph, uint16_t channel, uint16_t value)</code>
<b>Function descriptions</b>	configure channel output compare value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
<b>Input parameter{in}</b>	
<b>value</b>	0~65535

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPTIMER0 channel output compare value */
```

```
gptimer_channel_output_compare_value_config (GPTIMER0, GPTIMER_CH0, 0xff);
```

### **gptimer\_channel\_output\_additional\_compare\_value\_config**

The description of gptimer\_channel\_output\_additional\_compare\_value\_config is shown as below:

**Table 3-513. Function gptimer\_channel\_output\_additional\_compare\_value\_config**

Function name	gptimer_channel_output_additional_compare_value_config
Function prototype	void gptimer_channel_output_additional_compare_value_config(uint32_t timer_periph, uint16_t channel, uint16_t value)
Function descriptions	configure channel output additional compare value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	GPTIMER peripheral
GPTIMERx	x=0,1
Input parameter{in}	
channel	GPTIMER channel
GPTIMER_CH0	channel 0
GPTIMER_CH1	channel 1
Input parameter{in}	
value	0~65535
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPTIMER0 channel output additional compare value */
```

```
gptimer_channel_output_additional_compare_value_config (GPTIMER0, GPTIMER_CH0, 0xff);
```

### **gptimer\_channel\_output\_shadow\_config**

The description of gptimer\_channel\_output\_shadow\_config is shown as below:

Table 3-514. Function `gptimer_channel_output_shadow_config`

Function name	<code>gptimer_channel_output_shadow_config</code>
Function prototype	<code>void gptimer_channel_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t ocshadow)</code>
Function descriptions	configure channel output shadow function
Precondition	-
The called functions	-
Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
Input parameter{in}	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
Input parameter{in}	
<b>ocshadow</b>	channel output compare shadow
<i>GPTIMER_OC_SHADOW_ENABLE</i>	channel output compare shadow enable
<i>GPTIMER_OC_SHADOW_DISABLE</i>	channel output compare shadow disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPTIMER0 channel output shadow function */
```

```
gptimer_channel_output_shadow_config(GPTIMER0, GPTIMER_CH0,
GPTIMER_OC_SHADOW_ENABLE);
```

### **`gptimer_channel_output_additional_shadow_config`**

The description of `gptimer_channel_output_additional_shadow_config` is shown as below:

Table 3-515. Function `gptimer_channel_output_additional_shadow_config`

Function name	<code>gptimer_channel_output_additional_shadow_config</code>
Function prototype	<code>void gptimer_channel_output_additional_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t ocshadow)</code>
Function descriptions	configure channel output additional shadow function
Precondition	-
The called functions	-
Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1

Input parameter{in}	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
Input parameter{in}	
<b>ocshadow</b>	channel output compare shadow
<i>GPTIMER_OC_SHADOW_ENABLE</i>	channel output compare shadow enable
<i>GPTIMER_OC_SHADOW_DISABLE</i>	channel output compare shadow disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPTIMER0 channel output additional shadow function */
```

```
gptimer_channel_output_additional_shadow_config (GPTIMER0, GPTIMER_CH0,
GPTIMER_OC_SHADOW_ENABLE);
```

### **gptimer\_channel\_output\_control\_shadow\_config**

The description of gptimer\_channel\_output\_control\_shadow\_config is shown as below:

**Table 3-516. Function gptimer\_channel\_output\_control\_shadow\_config**

<b>Function name</b>	gptimer_channel_output_control_shadow_config
<b>Function prototype</b>	void gptimer_channel_output_control_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t ocshadow)
<b>Function descriptions</b>	configure channel output additional shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
Input parameter{in}	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
Input parameter{in}	
<b>ocshadow</b>	channel output compare shadow
<i>GPTIMER_OC_SHADOW_ENABLE</i>	channel output compare shadow enable
<i>GPTIMER_OC_SHADOW_DISABLE</i>	channel output compare shadow disable



Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure GPTIMER0 channel output additional shadow function */
```

```
gptimer_channel_output_control_shadow_config      (GPTIMER0,      GPTIMER_CH0,
GPTIMER_OC_SHADOW_ENABLE);
```

### **gptimer\_two\_channel\_output\_high\_check\_enable**

The description of gptimer\_two\_channel\_output\_high\_check\_enable is shown as below:

**Table 3-517. Function gptimer\_two\_channel\_output\_high\_check\_enable**

Function name	gptimer_two_channel_output_high_check_enable
Function prototype	void gptimer_two_channel_output_high_check_enable(uint32_t timer_periph)
Function descriptions	enable two channel simultaneous output high check function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	GPTIMER peripheral
GPTIMERx	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable GPTIMER0 two channel simultaneous output high check function */
```

```
gptimer_two_channel_output_high_check_enable (GPTIMER0);
```

### **gptimer\_two\_channel\_output\_high\_check\_disable**

The description of gptimer\_two\_channel\_output\_high\_check\_disable is shown as below:

**Table 3-518. Function gptimer\_two\_channel\_output\_high\_check\_disable**

Function name	gptimer_two_channel_output_high_check_disable
Function prototype	void gptimer_two_channel_output_high_check_disable(uint32_t timer_periph)
Function descriptions	disable two channel simultaneous output high check function
Precondition	-
The called functions	-

Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable GPTIMER0 two channel simultaneous output high check function */
```

```
gptimer_two_channel_output_high_check_disable (GPTIMER0);
```

### **gptimer\_two\_channel\_output\_low\_check\_enable**

The description of gptimer\_two\_channel\_output\_low\_check\_enable is shown as below:

**Table 3-519. Function gptimer\_two\_channel\_output\_low\_check\_enable**

<b>Function name</b>	gptimer_two_channel_output_low_check_enable
<b>Function prototype</b>	void gptimer_two_channel_output_low_check_enable(uint32_t timer_periph)
<b>Function descriptions</b>	enable two channel simultaneous output low check function
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable GPTIMER0 two channel simultaneous output low check function */
```

```
gptimer_two_channel_output_low_check_enable (GPTIMER0);
```

### **gptimer\_two\_channel\_output\_low\_check\_disable**

The description of gptimer\_two\_channel\_output\_low\_check\_disable is shown as below:

**Table 3-520. Function gptimer\_two\_channel\_output\_low\_check\_disable**

<b>Function name</b>	gptimer_two_channel_output_low_check_disable
<b>Function prototype</b>	void gptimer_two_channel_output_low_check_disable(uint32_t timer_periph)
<b>Function descriptions</b>	disable two channel simultaneous output low check function

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 two channel simultaneous output low check function */
```

```
gptimer_two_channel_output_low_check_disable (GPTIMER0);
```

### **gptimer\_period\_signal\_output\_enable**

The description of gptimer\_period\_signal\_output\_enable is shown as below:

**Table 3-521. Function gptimer\_period\_signal\_output\_enable**

<b>Function name</b>	gptimer_period_signal_output_enable
<b>Function prototype</b>	void gptimer_period_signal_output_enable(uint32_t timer_periph)
<b>Function descriptions</b>	enable counter cycle synchronization signal output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 counter cycle synchronization signal output */
```

```
gptimer_period_signal_output_enable (GPTIMER0);
```

### **gptimer\_period\_signal\_output\_disable**

The description of gptimer\_period\_signal\_output\_disable is shown as below:

**Table 3-522. Function gptimer\_period\_signal\_output\_disable**

<b>Function name</b>	gptimer_period_signal_output_disable
<b>Function prototype</b>	void gptimer_period_signal_output_disable(uint32_t timer_periph)
<b>Function descriptions</b>	disable counter cycle synchronization signal output

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 counter cycle synchronization signal output */
```

```
gptimer_period_signal_output_disable (GPTIMER0);
```

### **gptimer\_stop\_output\_set\_select**

The description of gptimer\_stop\_output\_set\_select is shown as below:

**Table 3-523. Function gptimer\_stop\_output\_set\_select**

<b>Function name</b>	gptimer_stop_output_set_select
<b>Function prototype</b>	void gptimer_stop_output_set_select(uint32_t timer_periph, uint32_t group)
<b>Function descriptions</b>	configure stop output source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>group</b>	stop output source select
<i>GPTIMER_OUTPUT_S TOP_GTOC0</i>	output stop GTOC0
<i>GPTIMER_OUTPUT_S TOP_GTOC1</i>	output stop GTOC1
<i>GPTIMER_OUTPUT_S TOP_GTOC2</i>	output stop GTOC2
<i>GPTIMER_OUTPUT_S TOP_GTOC3</i>	output stop GTOC3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 stop output source */
```

```
gptimer_stop_output_set_select (GPTIMER0, GPTIMER_OUTPUT_STOP_GTOC0);
```

### **gptimer\_stop\_output\_recover\_time\_select**

The description of gptimer\_stop\_output\_recover\_time\_select is shown as below:

**Table 3-524. Function gptimer\_stop\_output\_recover\_time\_select**

<b>Function name</b>	gptimer_stop_output_recover_time_select
<b>Function prototype</b>	void gptimer_stop_output_recover_time_select(uint32_t timer_periph, uint32_t time)
<b>Function descriptions</b>	output stop recover time select
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>time</b>	output stop recover time
<i>GPTIMER_OUTPUT_RECOVER_PERIOD_END</i>	output stop recover time is period end
<i>GPTIMER_OUTPUT_RECOVER_CREP_END</i>	output stop recover time is crep end
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* GPTIMER0 output stop recover time select */
```

```
gptimer_stop_output_recover_time_select (GPTIMER0,  
GPTIMER_OUTPUT_RECOVER_PERIOD_END);
```

### **gptimer\_channel\_complementary\_output\_struct\_para\_init**

The description of gptimer\_channel\_complementary\_output\_struct\_para\_init is shown as below:

**Table 3-525. Function gptimer\_channel\_complementary\_output\_struct\_para\_init**

<b>Function name</b>	gptimer_channel_complementary_output_struct_para_init
<b>Function prototype</b>	void gptimer_channel_complementary_output_struct_para_init(gptimer_com_o c_parameter_struct *comocpara)
<b>Function descriptions</b>	initialize GPTIMER complementary output parameter struct with a default

	value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>comocpara</b>	GPTIMER complementary output parameter struct, refer to <a href="#">Table 3-462. Structure gptimer_com_oc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* init parameter struct with a default value */
```

```
gptimer_com_oc_parameter_struct para;
```

```
gptimer_channel_complementary_output_struct_para_init (&para);
```

### **gptimer\_channel\_complementary\_output\_config**

The description of gptimer\_channel\_complementary\_output\_config is shown as below:

**Table 3-526. Function gptimer\_channel\_complementary\_output\_config**

<b>Function name</b>	gptimer_channel_complementary_output_config
<b>Function prototype</b>	void gptimer_channel_complementary_output_config(uint32_t timer_periph, gptimer_com_oc_parameter_struct *comocpara)
<b>Function descriptions</b>	configure GPTIMER channel complementary output function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<b>GPTIMERx</b>	x=0,1
<b>Input parameter{in}</b>	
<b>comocpara</b>	GPTIMER complementary output parameter struct, refer to <a href="#">Table 3-462. Structure gptimer_com_oc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 channel output function */
```

```
gptimer_com_oc_parameter_struct para;
```

```
gptimer_channel_complementary_output_config (GPTIMER0, GPTIMER_CH0, &para);
```

## gptimer\_channel\_io\_direction\_config

The description of gptimer\_channel\_io\_direction\_config is shown as below:

**Table 3-527. Function gptimer\_channel\_io\_direction\_config**

<b>Function name</b>	gptimer_channel_io_direction_config
<b>Function prototype</b>	void gptimer_channel_io_direction_config(uint32_t timer_periph, uint16_t channel, uint32_t io_direction)
<b>Function descriptions</b>	configure channel input or output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
<b>Input parameter{in}</b>	
<b>io_direction</b>	Input or output select
<i>GPTIMER_CHANNEL_OUTPUT</i>	channel output
<i>GPTIMER_CHANNEL_INPUT</i>	channel input
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure channel input */
```

```
gptimer_channel_io_direction_config(GPTIMER0, GPTIMER_CH0,
GPTIMER_CHANNEL_INPUT);
```

## gptimer\_channel\_io\_state\_config

The description of gptimer\_channel\_io\_state\_config is shown as below:

**Table 3-528. Function gptimer\_channel\_io\_state\_config**

<b>Function name</b>	gptimer_channel_io_state_config
<b>Function prototype</b>	void gptimer_channel_io_state_config(uint32_t timer_periph, uint16_t channel, uint32_t state)
<b>Function descriptions</b>	configure channel input or output enable state
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
Input parameter{in}	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
Input parameter{in}	
<b>state</b>	enable or disable
<i>GPTIMER_CHANNEL_ENABLE</i>	channel enable
<i>GPTIMER_CHANNEL_DISABLE</i>	channel disable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure channel input enable state */
```

```
gptimer_channel_io_state_config          (GPTIMER0,          GPTIMER_CH0,
gptimer_channel_io_state_config          GPTIMER_CHANNEL_ENABLE);
```

### **gptimer\_capture\_source\_struct\_para\_init**

The description of gptimer\_capture\_source\_struct\_para\_init is shown as below:

**Table 3-529. Function gptimer\_capture\_source\_struct\_para\_init**

<b>Function name</b>	gptimer_capture_source_struct_para_init
<b>Function prototype</b>	void gptimer_capture_source_struct_para_init(gptimer_capture_source_parameter_struct *counter_capture_source_para)
<b>Function descriptions</b>	initialize GPTIMER counter capture source init parameter struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>counter_capture_source_para</b>	counter capture source init parameter struct, refer to <a href="#">Table 3-458. Structure gptimer_capture_source_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-



Example:

```
/* init parameter struct with a default value */
gptimer_capture_source_parameter_struct para;
gptimer_capture_source_struct_para_init (&para);
```

### **gptimer\_capture\_source\_config**

The description of gptimer\_capture\_source\_config is shown as below:

**Table 3-530. Function gptimer\_capture\_source\_config**

Function name	gptimer_capture_source_config
Function prototype	void gptimer_capture_source_config(uint32_t timer_periph, uint16_t channel, gptimer_capture_source_parameter_struct *capture_source_para)
Function descriptions	initialize GPTIMER counter capture source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	GPTIMER peripheral
GPTIMERx	x=0,1
Input parameter{in}	
channel	GPTIMER channel
GPTIMER_CH0	channel 0
GPTIMER_CH1	channel 1
Input parameter{in}	
counter_capture_source_para	counter capture source init parameter struct, refer to <a href="#">Table 3-458. Structure gptimer_capture_source_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize GPTIMER0 counter capture source */
gptimer_capture_source_config (GPTIMER0, GPTIMER_CH0, capture_source_eti0);
```

### **gptimer\_channel\_input\_capture\_filter\_config**

The description of gptimer\_channel\_input\_capture\_filter\_config is shown as below:

**Table 3-531. Function gptimer\_channel\_input\_capture\_filter\_config**

Function name	gptimer_channel_input_capture_filter_config
Function prototype	void gptimer_channel_input_capture_filter_config(uint32_t timer_periph, uint32_t channel, uint32_t number)

<b>Function descriptions</b>	configure channel input capture filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
<b>Input parameter{in}</b>	
<b>number</b>	0~15
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 channel input capture filter */
```

```
gptimer_channel_input_capture_filter_config (GPTIMER0, GPTIMER_CH0, 0);
```

### **gptimer\_channel\_input\_capture\_prescaler\_config**

The description of gptimer\_channel\_input\_capture\_prescaler\_config is shown as below:

**Table 3-532. Function gptimer\_channel\_input\_capture\_prescaler\_config**

<b>Function name</b>	gptimer_channel_input_capture_prescaler_config
<b>Function prototype</b>	void gptimer_channel_input_capture_prescaler_config(uint32_t timer_periph, uint16_t channel, uint32_t prescaler)
<b>Function descriptions</b>	configure channel input capture prescaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
<b>Input parameter{in}</b>	
<b>prescaler</b>	prescaler value
<i>GPTIMER_IC_PSC_DIV1</i>	no prescaler
<i>GPTIMER_IC_PSC_DIV2</i>	divided by 2

V2	
<i>GPTIMER_IC_PSC_DIV1</i>	divided by 4
V4	
<i>GPTIMER_IC_PSC_DIV2</i>	divided by 8
V8	
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 channel input capture prescaler */
```

```
gptimer_channel_input_capture_prescaler_config (GPTIMER0, GPTIMER_CH0,
GPTIMER_IC_PSC_DIV1);
```

### **gptimer\_channel\_capture\_value\_register\_read**

The description of `gptimer_channel_capture_value_register_read` is shown as below:

**Table 3-533. Function `gptimer_channel_capture_value_register_read`**

<b>Function name</b>	<code>gptimer_channel_capture_value_register_read</code>
<b>Function prototype</b>	<code>uint32_t gptimer_channel_capture_value_register_read(uint32_t timer_periph, uint16_t channel)</code>
<b>Function descriptions</b>	read channel capture compare register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>channel</b>	GPTIMER channel
<i>GPTIMER_CH0</i>	channel 0
<i>GPTIMER_CH1</i>	channel 1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	capture value

Example:

```
/* read channel capture compare register value */
```

```
uint32_t value;
```

```
value = gptimer_channel_capture_value_register_read (GPTIMER0, GPTIMER_CH0);
```

## gptimer\_counter\_sync\_control\_select

The description of gptimer\_counter\_sync\_control\_select is shown as below:

**Table 3-534. Function gptimer\_counter\_sync\_control\_select**

<b>Function name</b>	gptimer_counter_sync_control_select
<b>Function prototype</b>	void gptimer_counter_sync_control_select(uint32_t timer_periph, uint32_t set)
<b>Function descriptions</b>	select counter sync reset set
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>group</b>	sync reset set
<i>GPTIMER_CNT_RESE T_GTOC0</i>	counter reset GTOC0
<i>GPTIMER_CNT_RESE T_GTOC1</i>	counter reset GTOC1
<i>GPTIMER_CNT_RESE T_GTOC2</i>	counter reset GTOC2
<i>GPTIMER_CNT_RESE T_GTOC3</i>	counter reset GTOC3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select GPTIMER0 counter sync reset set */
```

```
gptimer_counter_sync_control_select (GPTIMER0, GPTIMER_CNT_RESET_GTOC0);
```

## gptimer\_counter\_sync\_reset\_source\_select

The description of gptimer\_counter\_sync\_reset\_source\_select is shown as below:

**Table 3-535. Function gptimer\_counter\_sync\_reset\_source\_select**

<b>Function name</b>	gptimer_counter_sync_reset_source_select
<b>Function prototype</b>	void gptimer_counter_sync_reset_source_select(uint32_t timer_periph, uint32_t source)
<b>Function descriptions</b>	select counter sync reset source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>group</b>	sync reset source
<i>GPTIMER_SYNC_RES</i> <i>ET_CH0CV</i>	ch0cv as counter sync reset source
<i>GPTIMER_SYNC_RES</i> <i>ET_CH1CV</i>	ch1cv as counter sync reset source
<i>GPTIMER_SYNC_RES</i> <i>ET_CH0COMV_ADD</i>	ch0comval_add as counter sync reset source
<i>GPTIMER_SYNC_RES</i> <i>ET_CH1COMV_ADD</i>	ch1comval_add as counter sync reset source
<i>GPTIMER_SYNC_RES</i> <i>ET_UF</i>	overflow as counter sync reset source
<i>GPTIMER_SYNC_RES</i> <i>ET_OF</i>	underflow as counter sync reset source
<i>GPTIMER_SYNC_RES</i> <i>ET_CH</i>	channel input as counter sync reset source
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select GPTIMER0 counter sync reset source */
```

```
gptimer_counter_sync_reset_source_select          (GPTIMER0,  
GPTIMER_SYNC_RESET_CH0CV);
```

### **gptimer\_trigger\_adc\_compare\_enable**

The description of gptimer\_trigger\_adc\_compare\_enable is shown as below:

**Table 3-536. Function gptimer\_trigger\_adc\_compare\_enable**

<b>Function name</b>	gptimer_trigger_adc_compare_enable
<b>Function prototype</b>	void gptimer_trigger_adc_compare_enable(uint32_t timer_periph, uint32_t compare_time)
<b>Function descriptions</b>	enable compare event produce a A/D converter trigger signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>compare_time</b>	compare time

<i>GPTIMER_ADT1UEN</i>	enable counter up ADTCV1[15:0] match event as adc trigger signal
<i>GPTIMER_ADT1DEN</i>	enable counter down ADTCV1[15:0] match event as adc trigger signal
<i>GPTIMER_ADT2UEN</i>	enable counter up ADTCV2[15:0] match event as adc trigger signal
<i>GPTIMER_ADT2DEN</i>	enable counter down ADTCV2[15:0] match event as adc trigger signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 compare event produce a A/D converter trigger signal */
```

```
gptimer_trigger_adc_compare_enable (GPTIMER0, GPTIMER_ADT1UEN);
```

### **gptimer\_trigger\_adc\_compare\_disable**

The description of `gptimer_trigger_adc_compare_disable` is shown as below:

**Table 3-537. Function `gptimer_trigger_adc_compare_disable`**

<b>Function name</b>	<code>gptimer_trigger_adc_compare_disable</code>
<b>Function prototype</b>	<code>void gptimer_trigger_adc_compare_disable(uint32_t timer_periph, uint32_t compare_time)</code>
<b>Function descriptions</b>	disable compare event produce a A/D converter trigger signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>compare_time</b>	compare time
<i>GPTIMER_ADT1UEN</i>	enable counter up ADTCV1[15:0] match event as adc trigger signal
<i>GPTIMER_ADT1DEN</i>	enable counter down ADTCV1[15:0] match event as adc trigger signal
<i>GPTIMER_ADT2UEN</i>	enable counter up ADTCV2[15:0] match event as adc trigger signal
<i>GPTIMER_ADT2DEN</i>	enable counter down ADTCV2[15:0] match event as adc trigger signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 compare event produce a A/D converter trigger signal */
```

```
gptimer_trigger_adc_compare_disable (GPTIMER0, GPTIMER_ADT1UEN);
```

## gptimer\_trigger\_adc\_compare\_value\_config

The description of gptimer\_trigger\_adc\_compare\_value\_config is shown as below:

**Table 3-538. Function gptimer\_trigger\_adc\_compare\_value\_config**

<b>Function name</b>	gptimer_trigger_adc_compare_value_config
<b>Function prototype</b>	void gptimer_trigger_adc_compare_value_config(uint32_t timer_periph, uint32_t compare, uint16_t value)
<b>Function descriptions</b>	config trigger adc compare register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>compare</b>	compare select
<i>GPTIMER_ADC_COMPARE1</i>	select GPTIMERx_ADCCR1 ADTCV1
<i>GPTIMER_ADC_COMPARE2</i>	select GPTIMERx_ADCCR2 ADTCV2
<b>Input parameter{in}</b>	
<b>value</b>	0~65535
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config GPTIMER0 trigger adc compare register value */
```

```
gptimer_trigger_adc_compare_value_config (GPTIMER0, GPTIMER_ADC_COMPARE1, 0xff);
```

## gptimer\_trigger\_adc\_compare\_shadow\_enable

The description of gptimer\_trigger\_adc\_compare\_shadow\_enable is shown as below:

**Table 3-539. Function gptimer\_trigger\_adc\_compare\_shadow\_enable**

<b>Function name</b>	gptimer_trigger_adc_compare_shadow_enable
<b>Function prototype</b>	void gptimer_trigger_adc_compare_shadow_enable(uint32_t timer_periph, uint32_t compare)
<b>Function descriptions</b>	enable adc compare register shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral

<i>GPTIMERx</i>	<i>x</i> =0,1
<b>Input parameter{in}</b>	
<b>compare</b>	compare select
<i>GPTIMER_ADC_COMP</i> <i>ARE1</i>	select <i>GPTIMERx_ADCCR1</i> ADTCV1
<i>GPTIMER_ADC_COMP</i> <i>ARE2</i>	select <i>GPTIMERx_ADCCR2</i> ADTCV2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 adc compare regiseter shadow function */
```

```
gptimer_trigger_adc_compare_shadow_enable (GPTIMER0, GPTIMER_ADC_COMPARE1);
```

### **gptimer\_trigger\_adc\_compare\_shadow\_disable**

The description of `gptimer_trigger_adc_compare_shadow_disable` is shown as below:

**Table 3-540. Function `gptimer_trigger_adc_compare_shadow_disable`**

<b>Function name</b>	<code>gptimer_trigger_adc_compare_shadow_disable</code>
<b>Function prototype</b>	<code>void gptimer_trigger_adc_compare_shadow_disable(uint32_t timer_periph, uint32_t compare)</code>
<b>Function descriptions</b>	disable adc compare regiseter shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	<i>x</i> =0,1
<b>Input parameter{in}</b>	
<b>compare</b>	compare select
<i>GPTIMER_ADC_COMP</i> <i>ARE1</i>	select <i>GPTIMERx_ADCCR1</i> ADTCV1
<i>GPTIMER_ADC_COMP</i> <i>ARE2</i>	select <i>GPTIMERx_ADCCR2</i> ADTCV2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 adc compare regiseter shadow function */
```



```
gptimer_trigger_adc_compare_shadow_disable (GPTIMER0,
GPTIMER_ADC_COMPARE1);
```

### gptimer\_trigger\_adc\_adsm\_enable

The description of gptimer\_trigger\_adc\_adsm\_enable is shown as below:

**Table 3-541. Function gptimer\_trigger\_adc\_adsm\_enable**

<b>Function name</b>	gptimer_trigger_adc_adsm_enable
<b>Function prototype</b>	void gptimer_trigger_adc_adsm_enable(uint32_t timer_periph, uint32_t adsm)
<b>Function descriptions</b>	enable trigger adc adsm signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>adsm</b>	adsm signal select
<i>GPTIMER_ADCSM3</i>	adc monitor signal 3
<i>GPTIMER_ADCSM4</i>	adc monitor signal 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 trigger adc adsm signal */
```

```
gptimer_trigger_adc_adsm_enable (GPTIMER0, GPTIMER_ADCSM3);
```

### gptimer\_trigger\_adc\_adsm\_disable

The description of gptimer\_trigger\_adc\_adsm\_disable is shown as below:

**Table 3-542. Function gptimer\_trigger\_adc\_adsm\_disable**

<b>Function name</b>	gptimer_trigger_adc_adsm_disable
<b>Function prototype</b>	void gptimer_trigger_adc_adsm_disable(uint32_t timer_periph, uint32_t adsm)
<b>Function descriptions</b>	disable trigger adc adsm signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	

<b>adsm</b>	adsm signal select
<i>GPTIMER_ADCSM3</i>	adc monitor signal 3
<i>GPTIMER_ADCSM4</i>	adc monitor signal 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 trigger adc adsm signal */
```

```
gptimer_trigger_adc_adsm_disable (GPTIMER0, GPTIMER_ADCSM3);
```

### **gptimer\_trigger\_adc\_adsm\_select**

The description of `gptimer_trigger_adc_adsm_select` is shown as below:

**Table 3-543. Function `gptimer_trigger_adc_adsm_select`**

<b>Function name</b>	<code>gptimer_trigger_adc_adsm_select</code>
<b>Function prototype</b>	<code>void gptimer_trigger_adc_adsm_select(uint32_t timer_periph, uint32_t adsm, uint32_t trg_source)</code>
<b>Function descriptions</b>	select trigger adc adsm signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>adsm</b>	adsm signal select
<i>GPTIMER_ADCSM3</i>	adc monitor signal 3
<i>GPTIMER_ADCSM4</i>	adc monitor signal 4
<b>Input parameter{in}</b>	
<b>trg_source</b>	trigger source select
<i>GPTIMER_ADSM_ADT CV1</i>	adc monitor signal ADTCV1
<i>GPTIMER_ADSM_ADT CV2</i>	adc monitor signal ADTCV2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select GPTIMER0 trigger adc adsm signal */
```

```
gptimer_trigger_adc_adsm_select      (GPTIMER0,      GPTIMER_ADCSM3,
GPTIMER_ADSM_ADTCV1);
```

### gptimer\_trigger\_adc\_skipping\_config

The description of gptimer\_trigger\_adc\_skipping\_config is shown as below:

**Table 3-544. Function gptimer\_trigger\_adc\_skipping\_config**

<b>Function name</b>	gptimer_trigger_adc_skipping_config
<b>Function prototype</b>	void gptimer_trigger_adc_skipping_config(uint32_t timer_periph, uint32_t skipping_count, uint32_t value, uint32_t initial_value, uint32_t skipping_source)
<b>Function descriptions</b>	config trigger adc skipping value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>skipping_count</b>	skipping counter select
<i>GPTIMER_ADC_SKIP_COUNT1</i>	adc skipping counter 1
<i>GPTIMER_ADC_SKIP_COUNT2</i>	adc skipping counter 2
<b>Input parameter{in}</b>	
<b>value</b>	skipping value, 0~15
<b>Input parameter{in}</b>	
<b>initial_value</b>	initial value, 0~15
<b>Input parameter{in}</b>	
<b>skipping_source</b>	skipping source select
<i>GPTIMER_ADC_SKIP_NO</i>	no counting, no skipping
<i>GPTIMER_ADC_SKIP_ADTCV1</i>	the ADTCV1[15:0] comparison matching
<i>GPTIMER_ADC_SKIP_ADTCV2</i>	the ADTCV2[15:0] comparison matching
<i>GPTIMER_ADC_SKIP_ADTCV1_OR_ADTCV2</i>	the ADTCV1[15:0] or ADTCV2[15:0] comparison matching
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config GPTIMER0 trigger adc skipping value */
```

```
gptimer_trigger_adc_skipping_config (GPTIMER0, GPTIMER_ADC_SKIP_COUNT1, 2, 2,
GPTIMER_ADC_SKIP_ADTCV1);
```

### gptimer\_trigger\_adc\_skipping\_time\_select

The description of gptimer\_trigger\_adc\_skipping\_time\_select is shown as below:

**Table 3-545. Function gptimer\_trigger\_adc\_skipping\_time\_select**

Function name	gptimer_trigger_adc_skipping_time_select
Function prototype	void gptimer_trigger_adc_skipping_time_select(uint32_t timer_periph, uint32_t compare, uint32_t skipping_time)
Function descriptions	select trigger adc skipping function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	GPTIMER peripheral
GPTIMERx	x=0,1
Input parameter{in}	
compare	compare select
GPTIMER_ADC_COMPARE1	select GPTIMERx_ADCCR1 ADTCV1
GPTIMER_ADC_COMPARE2	select GPTIMERx_ADCCR2 ADTCV2
Input parameter{in}	
skipping_time	skipping time select
GPTIMER_ADC_REQ_NO	do not skipping
GPTIMER_ADC_REQ_CNT1_ZERO	adc req source when skipping counter 1 equal 0
GPTIMER_ADC_REQ_CNT2_ZERO	adc req source when skipping counter 2 equal 0
GPTIMER_ADC_REQ_CNT1_OR_CNT2_ZERO	adc req source when skipping counter 1/2 equal 0
GPTIMER_ADC_REQ_CNT1_SVAL	adc req source when skipping counter 1 equal skipping value
GPTIMER_ADC_REQ_CNT2_SVAL	adc req source when skipping counter 2 equal skipping value
GPTIMER_ADC_REQ_CNT1_OR_CNT2_SVAL	adc req source when skipping counter 1/2 equal skipping value
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* select GPTIMER0 trigger adc skipping function */
```

```
gptimer_trigger_adc_skipping_time_select (GPTIMER0, GPTIMER_ADC_COMPARE1,
GPTIMER_ADC_REQ_CNT1_ZERO);
```

### **gptimer\_trigger\_adc\_skipping\_counter\_read**

The description of gptimer\_trigger\_adc\_skipping\_counter\_read is shown as below:

**Table 3-546. Function gptimer\_trigger\_adc\_skipping\_counter\_read**

Function name	gptimer_trigger_adc_skipping_counter_read
Function prototype	uint16_t gptimer_trigger_adc_skipping_counter_read(uint32_t timer_periph, uint32_t skipping_count)
Function descriptions	read trigger adc skipping counter
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	GPTIMER peripheral
GPTIMERx	x=0,1
Input parameter{in}	
skipping_count	skipping counter select
GPTIMER_ADDINT_SK IP_COUNT1	additional interrupt skipping counter 1
GPTIMER_ADDINT_SK IP_COUNT2	additional interrupt skipping counter 2
Output parameter{out}	
-	-
Return value	
skipping counter	

Example:

```
/* read GPTIMER0 trigger adc skipping counter */
```

```
gptimer_trigger_adc_skipping_counter_read (GPTIMER0,
GPTIMER_ADDINT_SKIP_COUNT1);
```

### **gptimer\_additional\_interrupt\_skipping\_config**

The description of gptimer\_additional\_interrupt\_skipping\_config is shown as below:

**Table 3-547. Function gptimer\_additional\_interrupt\_skipping\_config**

Function name	gptimer_additional_interrupt_skipping_config
Function prototype	void gptimer_additional_interrupt_skipping_config(uint32_t timer_periph,

	uint32_t skipping_count, uint32_t value, uint32_t initial_value, uint32_t skipping_source)
<b>Function descriptions</b>	config additional interrupt skipping initial value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>skipping_count</b>	skipping counter select
<i>GPTIMER_ADDINT_SK IP_COUNT1</i>	additional interrupt skipping counter 1
<i>GPTIMER_ADDINT_SK IP_COUNT2</i>	additional interrupt skipping counter 2
<b>Input parameter{in}</b>	
<b>value</b>	skipping value, 0~15
<b>Input parameter{in}</b>	
<b>initial_value</b>	initial value, 0~15
<b>Input parameter{in}</b>	
<b>skipping_source</b>	skipping source select
<i>GPTIMER_ADDINT_SK IP_NO</i>	no additional interrupt skipping counter source
<i>GPTIMER_ADDINT_SK IP_OVERFLOW</i>	overflow as additional skipping counter source
<i>GPTIMER_ADDINT_SK IP_UNDERFLOW</i>	underflow as additional skipping counter source
<i>GPTIMER_ADDINT_SK IP_FLOW</i>	overflow and underflow as additional skipping counter source
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config GPTIMER0 additional interrupt skipping initial value */
```

```
gptimer_additional_interrupt_skipping_config(GPTIMER0,
GPTIMER_ADDINT_SKIP_COUNT1, 2, 2, GPTIMER_ADDINT_SKIP_OVERFLOW);
```

### **gptimer\_additional\_interrupt\_skipping\_time\_select**

The description of gptimer\_additional\_interrupt\_skipping\_time\_select is shown as below:

**Table 3-548. Function gptimer\_additional\_interrupt\_skipping\_time\_select**

<b>Function name</b>	gptimer_additional_interrupt_skipping_time_select
----------------------	---

<b>Function prototype</b>	void gptimer_additional_interrupt_skipping_time_select(uint32_t timer_periph, uint32_t event, uint32_t skipping_time)
<b>Function descriptions</b>	select additional interrupt skipping function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>event</b>	skipping event
<i>GPTIMER_ADDINT_SK IP_EVENT_CH0CV</i>	additional interrupt skipping event ch0cv
<i>GPTIMER_ADDINT_SK IP_EVENT_CH1CV</i>	additional interrupt skipping event ch1cv
<i>GPTIMER_ADDINT_SK IP_EVENT_CH0COMV _ADD</i>	additional interrupt skipping event ch0comv_add
<i>GPTIMER_ADDINT_SK IP_EVENT_CH1COMV _ADD</i>	additional interrupt skipping event ch1comv_add
<i>GPTIMER_ADDINT_SK IP_EVENT_UNDERFL OW</i>	additional interrupt skipping event underflow
<i>GPTIMER_ADDINT_SK IP_EVENT_OVERFLOW W</i>	additional interrupt skipping event overflow
<i>GPTIMER_ADDINT_SK IP_EVENT_ADTCV1</i>	additional interrupt skipping event adtcv1
<i>GPTIMER_ADDINT_SK IP_EVENT_ADTCV2</i>	additional interrupt skipping event adtcv2
<b>Input parameter{in}</b>	
<b>skipping_time</b>	skipping time
<i>GPTIMER_ADDINT_RE Q_NO</i>	do not skipping
<i>GPTIMER_ADDINT_RE Q_CNT1_ZERO</i>	additional interrupt req source when skipping counter 1 equal 0
<i>GPTIMER_ADDINT_RE Q_CNT2_ZERO</i>	additional interrupt req source when skipping counter 2 equal 0
<i>GPTIMER_ADDINT_RE Q_CNT1_OR_CNT2_Z ERO</i>	additional interrupt req source when skipping counter 1/2 equal 0
<i>GPTIMER_ADDINT_RE Q_CNT1_SVAL</i>	additional interrupt req source when skipping counter 1 equal skipping value

<i>GPTIMER_ADDINT_REQ_CNT2_SVAL</i>	additional interrupt req source when skipping counter 2 equal skipping value
<i>GPTIMER_ADDINT_REQ_CNT1_OR_CNT2_SVAL</i>	additional interrupt req source when skipping counter 1/2 equal skipping value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select GPTIMER0 additional interrupt skipping function */
```

```
gptimer_additional_interrupt_skipping_time_select (GPTIMER0,
GPTIMER_ADDINT_SKIP_EVENT_CH0CV, GPTIMER_ADDINT_REQ_CNT1_ZERO);
```

### **gptimer\_additional\_interrupt\_skipping\_counter\_read**

The description of `gptimer_additional_interrupt_skipping_counter_read` is shown as below:

**Table 3-549. Function `gptimer_additional_interrupt_skipping_counter_read`**

<b>Function name</b>	<code>gptimer_additional_interrupt_skipping_counter_read</code>
<b>Function prototype</b>	<code>uint16_t gptimer_additional_interrupt_skipping_counter_read(uint32_t timer_periph, uint32_t skipping_count)</code>
<b>Function descriptions</b>	read additional interrupt skipping counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>skipping_count</b>	skipping counter select
<i>GPTIMER_ADDINT_SKIP_COUNT1</i>	additional interrupt skipping counter 1
<i>GPTIMER_ADDINT_SKIP_COUNT2</i>	additional interrupt skipping counter 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>skipping counter</b>	

Example:

```
/* read GPTIMER0 additional interrupt skipping counter */
```

```
gptimer_additional_interrupt_skipping_counter_read (GPTIMER0,
GPTIMER_ADDINT_SKIP_COUNT1);
```



## gptimer\_flow\_interrupt\_skipping\_source\_select

The description of gptimer\_flow\_interrupt\_skipping\_source\_select is shown as below:

**Table 3-550. Function gptimer\_flow\_interrupt\_skipping\_source\_select**

<b>Function name</b>	gptimer_flow_interrupt_skipping_source_select
<b>Function prototype</b>	void gptimer_flow_interrupt_skipping_source_select(uint32_t timer_periph, uint32_t skipping_source)
<b>Function descriptions</b>	select flow interrupt skipping source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>skipping_source</b>	skipping soucre
<i>GPTIMER_FLOW_SKIP_SOURCE_NO</i>	no flow skipping
<i>GPTIMER_FLOW_SKIP_SOURCE_OVERFLOW</i>	overflow as flow skipping soucre
<i>GPTIMER_FLOW_SKIP_SOURCE_UNDERFLOW</i>	underflow as flow skipping soucre
<i>GPTIMER_FLOW_SKIP_SOURCE_FLOW</i>	overflow / underflow as flow skipping soucre
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select GPTIMER0 flow interrupt skipping source */
```

```
gptimer_flow_interrupt_skipping_source_select(GPTIMER0,  
GPTIMER_FLOW_SKIP_SOURCE_OVERFLOW);
```

## gptimer\_flow\_interrupt\_skipping\_link\_enable

The description of gptimer\_flow\_interrupt\_skipping\_link\_enable is shown as below:

**Table 3-551. Function gptimer\_flow\_interrupt\_skipping\_link\_enable**

<b>Function name</b>	gptimer_flow_interrupt_skipping_link_enable
<b>Function prototype</b>	void gptimer_flow_interrupt_skipping_link_enable(uint32_t timer_periph, uint32_t skipping_link_source)
<b>Function descriptions</b>	enable flow interrupt skipping link function

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>skipping_link_source</b>	skipping link source
<i>GPTIMER_SKIP_LINK_CH0CV</i>	interrupt skipping link channel 0 input capture/compare match
<i>GPTIMER_SKIP_LINK_CH1CV</i>	interrupt skipping link channel 1 input capture/compare match
<i>GPTIMER_SKIP_LINK_CH0COMV_ADD</i>	interrupt skipping link channel 0 additional value compare match
<i>GPTIMER_SKIP_LINK_CH1COMV_ADD</i>	interrupt skipping link channel 1 additional value compare match
<i>GPTIMER_SKIP_LINK_ADCCR1</i>	interrupt skipping link trigger signal generated by ADTCV1[15:0]
<i>GPTIMER_SKIP_LINK_ADCCR2</i>	interrupt skipping link trigger signal generated by ADTCV2[15:0]
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GPTIMER0 flow interrupt skipping link function */
```

```
gptimer_flow_interrupt_skipping_link_enable (GPTIMER0, GPTIMER_SKIP_LINK_CH0CV);
```

### **gptimer\_flow\_interrupt\_skipping\_link\_disable**

The description of gptimer\_flow\_interrupt\_skipping\_link\_disable is shown as below:

**Table 3-552. Function gptimer\_flow\_interrupt\_skipping\_link\_disable**

<b>Function name</b>	gptimer_flow_interrupt_skipping_link_disable
<b>Function prototype</b>	void gptimer_flow_interrupt_skipping_link_disable(uint32_t timer_periph, uint32_t skipping_link_source)
<b>Function descriptions</b>	disable flow interrupt skipping link function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>skipping_link_source</b>	skipping link source

<i>GPTIMER_SKIP_LINK_CH0CV</i>	interrupt skipping link channel 0 input capture/compare match
<i>GPTIMER_SKIP_LINK_CH1CV</i>	interrupt skipping link channel 1 input capture/compare match
<i>GPTIMER_SKIP_LINK_CH0COMV_ADD</i>	interrupt skipping link channel 0 additional value compare match
<i>GPTIMER_SKIP_LINK_CH1COMV_ADD</i>	interrupt skipping link channel 1 additional value compare match
<i>GPTIMER_SKIP_LINK_ADCCR1</i>	interrupt skipping link trigger signal generated by ADTCV1[15:0]
<i>GPTIMER_SKIP_LINK_ADCCR2</i>	interrupt skipping link trigger signal generated by ADTCV2[15:0]
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GPTIMER0 flow interrupt skipping link function */
```

```
gptimer_flow_interrupt_skipping_link_disable (GPTIMER0, GPTIMER_SKIP_LINK_CH0CV);
```

### **gptimer\_flow\_interrupt\_skipping\_num\_config**

The description of `gptimer_flow_interrupt_skipping_num_config` is shown as below:

**Table 3-553. Function `gptimer_flow_interrupt_skipping_num_config`**

<b>Function name</b>	<code>gptimer_flow_interrupt_skipping_num_config</code>
<b>Function prototype</b>	<code>void gptimer_flow_interrupt_skipping_num_config(uint32_t timer_periph, uint32_t number)</code>
<b>Function descriptions</b>	configure flow interrupt repetition skipping number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>number</b>	the interrupt skipping value, 0~7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 flow interrupt repetition skipping number */
```

`gptimer_flow_interrupt_skipping_num_config (GPTIMER0, 7);`

### **gptimer\_flow\_interrupt\_skipping\_counter\_read**

The description of `gptimer_flow_interrupt_skipping_counter_read` is shown as below:

**Table 3-554. Function `gptimer_flow_interrupt_skipping_counter_read`**

<b>Function name</b>	<code>gptimer_flow_interrupt_skipping_counter_read</code>
<b>Function prototype</b>	<code>uint16_t gptimer_flow_interrupt_skipping_counter_read(uint32_t timer_periph)</code>
<b>Function descriptions</b>	read flow interrupt skipping counter value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	skipping counter

Example:

```
/* read GPTIMER0 flow interrupt skipping counter value */
```

```
uint16_t value;
```

```
value = gptimer_flow_interrupt_skipping_counter_read (GPTIMER0);
```

### **gptimer\_write\_chxval\_register\_config**

The description of `gptimer_write_chxval_register_config` is shown as below:

**Table 3-555. Function `gptimer_write_chxval_register_config`**

<b>Function name</b>	<code>gptimer_write_chxval_register_config</code>
<b>Function prototype</b>	<code>void gptimer_write_chxval_register_config(uint32_t timer_periph, uint16_t ccsel)</code>
<b>Function descriptions</b>	configure write CHxVAL register selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>ccsel</b>	write CHxVAL register protect enable or disable
<i>GPTIMER_CHVSEL_DISABLE</i>	no effect

<i>GPTIMER_CHVSEL_ENABLE</i>	when write the CHxVAL register, if the write value is same as the CHxVAL value, the write access is ignored
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GPTIMER0 write CHxVAL register selection */
```

```
gptimer_write_chxval_register_config (GPTIMER0, GPTIMER_CHVSEL_ENABLE);
```

### **gptimer\_flag\_get**

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

**Table 3-556. Function gptimer\_flag\_get**

<b>Function name</b>	gptimer_flag_get
<b>Function prototype</b>	FlagStatus gptimer_flag_get(uint32_t timer_periph, uint32_t flag)
<b>Function descriptions</b>	get GPTIMER flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>flag</b>	the GPTIMER flags
<i>GPTIMER_FLAG_CH0</i>	GPTIMER channel 0 capture or compare flag
<i>GPTIMER_FLAG_CH1</i>	GPTIMER channel 1 capture or compare flag
<i>GPTIMER_FLAG_OVERFLOW</i>	GPTIMER counter overflow flag
<i>GPTIMER_FLAG_UNDERFLOW</i>	GPTIMER counter underflow flag
<i>GPTIMER_FLAG_CH0_COMADD</i>	GPTIMER additional channel 0 compare flag
<i>GPTIMER_FLAG_CH1_COMADD</i>	GPTIMER additional channel 1 compare flag
<i>GPTIMER_FLAG_UP</i>	GPTIMER counter reset flag
<i>GPTIMER_FLAG_ADT1CMU</i>	GPTIMER adc trigger 1 up count compare match flag
<i>GPTIMER_FLAG_ADT1CMD</i>	GPTIMER adc trigger 1 down count compare match flag
<i>GPTIMER_FLAG_ADT2CMU</i>	GPTIMER adc trigger 2 up count compare match flag
<i>GPTIMER_FLAG_ADT2CMD</i>	GPTIMER adc trigger 2 down count compare match flag

<i>2CMD</i>	
<i>GPTIMER_FLAG_CHH OUT</i>	GPTIMER output simultaneous high error flag
<i>GPTIMER_FLAG_CHL OUT</i>	GPTIMER output simultaneous low error flag
<i>GPTIMER_FLAG_RCE ND</i>	GPTIMER repetition count end flag
<i>GPTIMER_FLAG_DIR</i>	GPTIMER counter direction flag
<i>GPTIMER_FLAG_OST</i>	GPTIMER output stop flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus: SET or RESET</b>	

Example:

```
/* get GPTIMER0 flags */
```

```
gptimer_flag_get (GPTIMER0, GPTIMER_FLAG_CH0);
```

### **gptimer\_flag\_clear**

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

**Table 3-557. Function gptimer\_flag\_clear**

<b>Function name</b>	gptimer_flag_clear
<b>Function prototype</b>	void gptimer_flag_clear(uint32_t timer_periph, uint32_t flag)
<b>Function descriptions</b>	reset GPTIMER flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>flag</b>	the GPTIMER flags
<i>GPTIMER_FLAG_CH0</i>	GPTIMER channel 1 capture or compare flag
<i>GPTIMER_FLAG_CH1</i>	GPTIMER channel 2 capture or compare flag
<i>GPTIMER_FLAG_OVE RFLOW</i>	GPTIMER counter overflow flag
<i>GPTIMER_FLAG_UND ERFLOW</i>	GPTIMER counter underflow flag
<i>GPTIMER_FLAG_CH0 COMADD</i>	GPTIMER additional channel 1 compare flag
<i>GPTIMER_FLAG_CH1 COMADD</i>	GPTIMER additional channel 2 compare flag

<i>GPTIMER_FLAG_UP</i>	GPTIMER counter reset flag
<i>GPTIMER_FLAG_ADT1CMU</i>	GPTIMER adc trigger 1 up count compare match flag
<i>GPTIMER_FLAG_ADT1CMD</i>	GPTIMER adc trigger 1 down count compare match flag
<i>GPTIMER_FLAG_ADT2CMU</i>	GPTIMER adc trigger 2 up count compare match flag
<i>GPTIMER_FLAG_ADT2CMD</i>	GPTIMER adc trigger 2 down count compare match flag
<i>GPTIMER_FLAG_CHHOUT</i>	GPTIMER output simultaneous high error flag
<i>GPTIMER_FLAG_CHLOUT</i>	GPTIMER output simultaneous low error flag
<i>GPTIMER_FLAG_RCE</i> <i>ND</i>	GPTIMER repetition count end flag
<i>GPTIMER_FLAG_DIR</i>	GPTIMER counter direction flag
<i>GPTIMER_FLAG_OST</i>	GPTIMER output stop flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset GPTIMER0 flags */
```

```
gptimer_flag_clear (GPTIMER0, GPTIMER_FLAG_CH0);
```

### **gptimer\_interrupt\_enable**

The description of `gptimer_interrupt_enable` is shown as below:

**Table 3-558. Function `gptimer_interrupt_enable`**

<b>Function name</b>	<code>gptimer_interrupt_enable</code>
<b>Function prototype</b>	<code>void gptimer_interrupt_enable(uint32_t timer_periph, uint32_t interrupt)</code>
<b>Function descriptions</b>	enable the GPTIMER interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>interrupt</b>	GPTIMER interrupt enable source
<i>GPTIMER_INT_CH0</i>	GPTIMER channel 0 capture or compare interrupt
<i>GPTIMER_INT_CH1</i>	GPTIMER channel 1 capture or compare interrupt
<i>GPTIMER_INT_OVERF</i>	GPTIMER counter overflow interrupt

<i>LOW</i>	
<i>GPTIMER_INT_UNDE RFLOW</i>	GPTIMER counter underflow interrupt
<i>GPTIMER_INT_CH0CO MADD</i>	GPTIMER additional channel 0 compare interrupt
<i>GPTIMER_INT_CH1CO MADD</i>	GPTIMER additional channel 1 compare interrupt
<i>GPTIMER_INT_UP</i>	GPTIMER counter reset interrupt
<i>GPTIMER_INT_CREP</i>	GPTIMER repetition count end interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the GPTIMER0 interrupt */
```

```
gptimer_interrupt_enable (GPTIMER0, GPTIMER_INT_CH0);
```

### **gptimer\_interrupt\_disable**

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

**Table 3-559. Function gptimer\_interrupt\_disable**

<b>Function name</b>	gptimer_interrupt_disable
<b>Function prototype</b>	void gptimer_interrupt_disable(uint32_t timer_periph, uint32_t interrupt)
<b>Function descriptions</b>	disable the GPTIMER interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>interrupt</b>	GPTIMER interrupt enable source
<i>GPTIMER_INT_CH0</i>	GPTIMER channel 0 capture or compare interrupt
<i>GPTIMER_INT_CH1</i>	GPTIMER channel 1 capture or compare interrupt
<i>GPTIMER_INT_OVERFLOW LOW</i>	GPTIMER counter overflow interrupt
<i>GPTIMER_INT_UNDE RFLOW</i>	GPTIMER counter underflow interrupt
<i>GPTIMER_INT_CH0CO MADD</i>	GPTIMER additional channel 0 compare interrupt
<i>GPTIMER_INT_CH1CO MADD</i>	GPTIMER additional channel 1 compare interrupt
<i>GPTIMER_INT_UP</i>	GPTIMER counter reset interrupt



<i>GPTIMER_INT_CREP</i>	GPTIMER repetition count end interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the GPTIMER0 interrupt */
```

```
gptimer_interrupt_disable (GPTIMER0, GPTIMER_INT_CH0);
```

### **gptimer\_interrupt\_flag\_get**

The description of `gptimer_interrupt_flag_get` is shown as below:

**Table 3-560. Function `gptimer_interrupt_flag_get`**

<b>Function name</b>	<code>gptimer_interrupt_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus gptimer_interrupt_flag_get(uint32_t timer_periph, uint32_t int_flag)</code>
<b>Function descriptions</b>	get GPTIMER interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	GPTIMER peripheral
<i>GPTIMERx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>int_flag</b>	the GPTIMER interrupt flags
<i>GPTIMER_INT_FLAG_CH0</i>	GPTIMER channel 0 capture or compare interrupt flag
<i>GPTIMER_INT_FLAG_CH1</i>	GPTIMER channel 1 capture or compare interrupt flag
<i>GPTIMER_INT_FLAG_OVERFLOW</i>	GPTIMER counter overflow interrupt flag
<i>GPTIMER_INT_FLAG_UNDERFLOW</i>	GPTIMER counter underflow interrupt flag
<i>GPTIMER_INT_FLAG_CH0COMADD</i>	GPTIMER additional channel 0 compare interrupt flag
<i>GPTIMER_INT_FLAG_CH1COMADD</i>	GPTIMER additional channel 1 compare interrupt flag
<i>GPTIMER_INT_FLAG_UP</i>	GPTIMER counter reset interrupt flag
<i>GPTIMER_INT_FLAG_CREP</i>	GPTIMER repetition count end interrupt flag
<b>Output parameter{out}</b>	
-	-

Return value	
FlagStatus: SET or RESET	

Example:

```
/* get GPTIMER0 interrupt flag */
```

```
gptimer_interrupt_flag_get (GPTIMER0, GPTIMER_INT_FLAG_CH0);
```

### **gptimer\_interrupt\_flag\_clear**

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

**Table 3-561. Function gptimer\_interrupt\_flag\_clear**

Function name	gptimer_interrupt_flag_clear
Function prototype	void gptimer_interrupt_flag_clear(uint32_t timer_periph, uint32_t int_flag)
Function descriptions	clear GPTIMER interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	GPTIMER peripheral
GPTIMERx	x=0,1
Input parameter{in}	
int_flag	the GPTIMER interrupt flags
GPTIMER_INT_FLAG_CH0	GPTIMER channel 0 capture or compare interrupt flag
GPTIMER_INT_FLAG_CH1	GPTIMER channel 1 capture or compare interrupt flag
GPTIMER_INT_FLAG_OVERFLOW	GPTIMER counter overflow interrupt flag
GPTIMER_INT_FLAG_UNDERFLOW	GPTIMER counter underflow interrupt flag
GPTIMER_INT_FLAG_CH0COMADD	GPTIMER additional channel 0 compare interrupt flag
GPTIMER_INT_FLAG_CH1COMADD	GPTIMER additional channel 1 compare interrupt flag
GPTIMER_INT_FLAG_UP	GPTIMER counter clear interrupt flag
GPTIMER_INT_FLAG_CREP	GPTIMER repetition count end interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear GPTIMER0 interrupt flag */
```

```
gptimer_interrupt_flag_clear (GPTIMER0, GPTIMER_INT_FLAG_CH0);
```

## 3.18. GTOC

The GTOC can generate request to close the output pin of GPTIMER. The closed pins can output high-impedance (Hi-Z) state. Output closing request sources can come from different modules on the chip, such as CMP, GPTIMER and so on. The GTOC registers are listed in chapter [3.18.1](#), the GTOC firmware functions are introduced in chapter [3.18.2](#).

### 3.18.1. Descriptions of Peripheral registers

GTOC registers are listed in the table shown as below:

**Table 3-562. GTOC registers**

Register	Descriptions
GTOCx_CFG	GTOCx configuration register
GTOCx_OCRCTL	GTOCx output closing request control register
GTOCx_WP	GTOCx write protection register
GTOCx_ECRCTL	GTOCx extended closing request control register

### 3.18.2. Descriptions of Peripheral functions

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

**Table 3-563. GTOC firmware function**

Function name	Function description
gtoc_deinit	deinitialize GTOC
gtoc_output_closing_request_enable	enable GTOC output closing request
gtoc_gptimer_trigger_status_get	get GPTIMER external trigger input status
gtoc_software_request_generate	generate software output closing request
gtoc_software_request_stop	stop software output closing request
gtoc_input_detection_mode_select	select GTOCx_IN input detection mode
gtoc_input_polarity_config	configure GTOCx_IN input polarity
gtoc_digital_filter_enable	enable GTOC digital filter
gtoc_digital_filter_disable	disable GTOC digital filter
gtoc_digital_filter_config	configure GTOC digital filter
gtoc_output_closing_request_mask	mask GTOC output closing request
gtoc_register_write_enable	enable GTOC extended closing request control register write
gtoc_register_write_disable	disable GTOC extended closing request control register write
gtoc_extended_closing_request_enable	enable extended closing request

gtoc_extended_closing_request_disable	disable extended closing request
gtoc_extended_closing_request_config	configure extended closing request
gtoc_extended_closing_request_mask	mask GTOC extended closing request
gtoc_flag_get	get GTOC flag
gtoc_flag_clear	clear GTOC flag
gtoc_interrupt_flag_get	get GTOC interrupt flag
gtoc_interrupt_flag_clear	clear GTOC interrupt flag

## gtoc\_deinit

The description of gtoc\_deinit is shown as below:

**Table 3-564. Function gtoc\_deinit**

<b>Function name</b>	gtoc_deinit
<b>Function prototype</b>	void gtoc_deinit(uint32_t gtoc_periph)
<b>Function descriptions</b>	deinitialize GTOC
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize GTOC0 */
gtoc_deinit(GTOC0);
```

## gtoc\_output\_closing\_request\_enable

The description of gtoc\_output\_closing\_request\_enable is shown as below:

**Table 3-565. Function gtoc\_output\_closing\_request\_enable**

<b>Function name</b>	gtoc_output_closing_request_enable
<b>Function prototype</b>	void gtoc_output_closing_request_enable(uint32_t gtoc_periph, uint16_t gtoc_ocr_source)
<b>Function descriptions</b>	enable GTOC output closing request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	

<b>gtoc_ocr_source</b>	gtoc output closing request source
<i>GTOC_OCR_SOURCE_GTOCPIN</i>	GTOCx_IN pin input detection
<i>GTOC_OCR_SOURCE_GPTIMER</i>	GPTIMER output fault detection
<i>GTOC_OCR_SOURCE_HXTALSTUCK</i>	HXTAL stuck detection enable
<i>GTOC_OCR_SOURCE_LOCKUP</i>	CPU LOCKUP detection
<i>GTOC_OCR_SOURCE_CMP0</i>	CMP0 valid edge detection
<i>GTOC_OCR_SOURCE_CMP1</i>	CMP1 valid edge detection
<i>GTOC_OCR_SOURCE_CMP2</i>	CMP2 valid edge detection
<i>GTOC_OCR_SOURCE_CMP3</i>	CMP3 valid edge detection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GTOC0 output closing request generated by GTOCx_IN pin input detection */
gtoc_output_closing_request_enable(GTOC0, GTOC_OCR_SOURCE_GTOCPIN);
```

### gtoc\_gptimer\_trigger\_status\_get

The description of gtoc\_gptimer\_trigger\_status\_get is shown as below:

**Table 3-566. Function gtoc\_gptimer\_trigger\_status\_get**

<b>Function name</b>	gtoc_gptimer_trigger_status_get
<b>Function prototype</b>	uint32_t gtoc_gptimer_trigger_status_get(uint32_t gtoc_periph)
<b>Function descriptions</b>	get GPTIMER external trigger input status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the trigger input status
<i>GPTIMER_TRIGGER_S</i>	trigger input is high

<i>TATUS_HIGH</i>	
<i>GPTIMER_TRIGGER_S</i>	trigger input is low
<i>TATUS_LOW</i>	

Example:

```
uint32_t status;
```

```
/* get GPTIMER external trigger input status from GTOC0 */
```

```
status = gtoc_gptimer_trigger_status_get(GTOC0);
```

### gtoc\_software\_request\_generate

The description of gtoc\_software\_request\_generate is shown as below:

**Table 3-567. Function gtoc\_software\_request\_generate**

<b>Function name</b>	gtoc_software_request_generate
<b>Function prototype</b>	void gtoc_software_request_generate(uint32_t gtoc_periph)
<b>Function descriptions</b>	generate software output closing request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* generate software output closing request from GTOC0 */
```

```
gtoc_software_request_generate(GTOC0);
```

### gtoc\_software\_request\_stop

The description of gtoc\_software\_request\_stop is shown as below:

**Table 3-568. Function gtoc\_software\_request\_stop**

<b>Function name</b>	gtoc_software_request_stop
<b>Function prototype</b>	void gtoc_software_request_stop(uint32_t gtoc_periph)
<b>Function descriptions</b>	stop software output closing request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop software output closing request from GTOC0 */
```

```
gtoc_software_request_stop(GTOC0);
```

### gtoc\_input\_detection\_mode\_select

The description of gtoc\_input\_detection\_mode\_select is shown as below:

**Table 3-569. Function gtoc\_input\_detection\_mode\_select**

Function name	gtoc_input_detection_mode_select
Function prototype	void gtoc_input_detection_mode_select(uint32_t gtoc_periph, uint32_t detection_mode)
Function descriptions	select GTOCx_IN input detection mode
Precondition	-
The called functions	-
Input parameter{in}	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
Input parameter{in}	
<b>detection_mode</b>	GTOCx_IN input detection mode
<i>GTOC_INPUT_DETECTI ON_LEVEL</i>	level detection
<i>GTOC_INPUT_DETECTI ON_EDGE</i>	edge detection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select GTOC0_IN input detection mode */
```

```
gtoc_input_detection_mode_select(GTOC0, GTOC_INPUT_DETECTION_LEVEL);
```

### gtoc\_input\_polarity\_config

The description of gtoc\_input\_polarity\_config is shown as below:

**Table 3-570. Function gtoc\_input\_polarity\_config**

Function name	gtoc_input_polarity_config
---------------	----------------------------

<b>Function prototype</b>	void gtoc_input_polarity_config(uint32_t gtoc_periph, uint32_t input_polarity)
<b>Function descriptions</b>	configure GTOCx_IN input polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>input_polarity</b>	GTOCx_IN input polarity
<i>GTOC_INPUT_POLARITY_NONINVERTED</i>	GTOCx_IN input not inverted
<i>GTOC_INPUT_DETECTI ON_EDGE</i>	GTOCx_IN input inverted
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GTOC0_IN input polarity */
```

```
gtoc_input_polarity_config(GTOC0, GTOC_INPUT_POLARITY_NONINVERTED);
```

### gtoc\_digital\_filter\_enable

The description of gtoc\_digital\_filter\_enable is shown as below:

**Table 3-571. Function gtoc\_input\_polarity\_config**

<b>Function name</b>	gtoc_digital_filter_enable
<b>Function prototype</b>	void gtoc_digital_filter_enable(uint32_t gtoc_periph)
<b>Function descriptions</b>	enable GTOC digital filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GTOC0 digital filter */
```



```
gtoc_digital_filter_enable(GTOC0);
```

### gtoc\_digital\_filter\_disable

The description of gtoc\_digital\_filter\_disable is shown as below:

**Table 3-572. Function gtoc\_digital\_filter\_disable**

<b>Function name</b>	gtoc_digital_filter_disable
<b>Function prototype</b>	void gtoc_digital_filter_disable(uint32_t gtoc_periph)
<b>Function descriptions</b>	disable GTOC digital filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GTOC0 digital filter */
gtoc_digital_filter_disable(GTOC0);
```

### gtoc\_digital\_filter\_config

The description of gtoc\_digital\_filter\_config is shown as below:

**Table 3-573. Function gtoc\_digital\_filter\_config**

<b>Function name</b>	gtoc_digital_filter_config
<b>Function prototype</b>	void gtoc_digital_filter_config(uint32_t gtoc_periph, gtoc_filter_sampling_freq_enum sampling_frequency, uint32_t sampling_number)
<b>Function descriptions</b>	configure GTOC digital filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>sampling_frequency</b>	GTOCx digital filter sampling frequency
<i>GTOC_SAMPLING_FREQUENCY_DIV1</i>	GTOC digital filter sampling frequency is $f_{HCLK}/1$
<i>GTOC_SAMPLING_FREQUENCY_DIV8</i>	GTOC digital filter sampling frequency is $f_{HCLK}/8$

<i>GTOC_SAMPLING_FREQUENCY_DIV32</i>	GTOC digital filter sampling frequency is $f_{HCLK}/32$
<i>GTOC_SAMPLING_FREQUENCY_DIV128</i>	GTOC digital filter sampling frequency is $f_{HCLK}/128$
<i>GTOC_SAMPLING_FREQUENCY_DIV2</i>	GTOC digital filter sampling frequency is $f_{HCLK}/2$
<i>GTOC_SAMPLING_FREQUENCY_DIV4</i>	GTOC digital filter sampling frequency is $f_{HCLK}/4$
<i>GTOC_SAMPLING_FREQUENCY_DIV16</i>	GTOC digital filter sampling frequency is $f_{HCLK}/16$
<i>GTOC_SAMPLING_FREQUENCY_DIV64</i>	GTOC digital filter sampling frequency is $f_{HCLK}/64$
<i>GTOC_SAMPLING_FREQUENCY_DIV256</i>	GTOC digital filter sampling frequency is $f_{HCLK}/256$
<i>GTOC_SAMPLING_FREQUENCY_DIV512</i>	GTOC digital filter sampling frequency is $f_{HCLK}/512$
<b>Input parameter{in}</b>	
<b>sampling_number</b>	GTOCx digital filter sampling number
<i>GTOC_SAMPLING_NUM_3_TIMES</i>	GTOC digital filter sampling number is three times
<i>GTOC_SAMPLING_NUM_4_TIMES</i>	GTOC digital filter sampling number is four times
<i>GTOC_SAMPLING_NUM_5_TIMES</i>	GTOC digital filter sampling number is five times
<i>GTOC_SAMPLING_NUM_6_TIMES</i>	GTOC digital filter sampling number is six times
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure GTOC0 digital filter */
```

```
gtoc_digital_filter_config(GTOC0, GTOC_SAMPLING_FREQUENCY_DIV1,
GTOC_SAMPLING_NUM_3_TIMES);
```

### **gtoc\_output\_closing\_request\_mask**

The description of gtoc\_output\_closing\_request\_mask is shown as below:

**Table 3-574. Function gtoc\_output\_closing\_request\_mask**

<b>Function name</b>	gtoc_output_closing_request_mask
<b>Function prototype</b>	void gtoc_output_closing_request_mask(uint32_t gtoc_periph, uint32_t mask_source)

<b>Function descriptions</b>	mask GTOC output closing request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>mask_source</b>	mask source for output closing request
<i>GTOC_OCRMKSEL_NO T_MASKED</i>	Output closing request is not masked
<i>GTOC_OCRMKSEL_GP TIMER0_CH0</i>	Output closing request is masked by GPTIMER0_CH0
<i>GTOC_OCRMKSEL_GP TIMER0_CH1</i>	Output closing request is masked by GPTIMER0_CH1
<i>GTOC_OCRMKSEL_GP TIMER1_CH0</i>	Output closing request is masked by GPTIMER1_CH0
<i>GTOC_OCRMKSEL_GP TIMER1_CH1</i>	Output closing request is masked by GPTIMER1_CH1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* mask GTOC output closing request */
```

```
gtoc_output_closing_request_mask(GTOC0, GTOC_OCRMKSEL_GPTIMER0_CH0);
```

### gtoc\_register\_write\_enable

The description of gtoc\_register\_write\_enable is shown as below:

**Table 3-575. Function gtoc\_register\_write\_enable**

<b>Function name</b>	gtoc_register_write_enable
<b>Function prototype</b>	void gtoc_register_write_enable(uint32_t gtoc_periph)
<b>Function descriptions</b>	enable GTOC extended closing request control register write
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable GTOC0 extended closing request control register write */
gtoc_register_write_enable(GTOC0);
```

### gtoc\_register\_write\_disable

The description of gtoc\_register\_write\_disable is shown as below:

**Table 3-576. Function gtoc\_register\_write\_disable**

<b>Function name</b>	gtoc_register_write_disable
<b>Function prototype</b>	void gtoc_register_write_disable(uint32_t gtoc_periph)
<b>Function descriptions</b>	disable GTOC extended closing request control register write
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<b>GTOCx</b>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable GTOC0 extended closing request control register write */
gtoc_register_write_disable(GTOC0);
```

### gtoc\_extended\_closing\_request\_enable

The description of gtoc\_extended\_closing\_request\_enable is shown as below:

**Table 3-577. Function gtoc\_extended\_closing\_request\_enable**

<b>Function name</b>	gtoc_extended_closing_request_enable
<b>Function prototype</b>	void gtoc_extended_closing_request_enable(uint32_t gtoc_periph)
<b>Function descriptions</b>	enable extended closing request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<b>GTOCx</b>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable extended closing request from GTOC0 */
gtoc_extended_closing_request_enable(GTOC0);
```

### gtoc\_extended\_closing\_request\_disable

The description of gtoc\_extended\_closing\_request\_disable is shown as below:

**Table 3-578. Function gtoc\_extended\_closing\_request\_disable**

<b>Function name</b>	gtoc_extended_closing_request_disable
<b>Function prototype</b>	void gtoc_extended_closing_request_disable(uint32_t gtoc_periph)
<b>Function descriptions</b>	disable extended closing request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable extended closing request from GTOC0 */
gtoc_extended_closing_request_disable(GTOC0);
```

### gtoc\_extended\_closing\_request\_config

The description of gtoc\_extended\_closing\_request\_config is shown as below:

**Table 3-579. Function gtoc\_extended\_closing\_request\_config**

<b>Function name</b>	gtoc_extended_closing_request_config
<b>Function prototype</b>	void gtoc_extended_closing_request_config(uint32_t gtoc_periph, gtoc_ext_closing_req_source_enum gtoc_ecr_source, uint32_t valid_level)
<b>Function descriptions</b>	configure extended closing request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>gtoc_ecr_source</b>	GTOC extended closing request source
<i>GTOC_ECR_SOURCE_</i>	CMP0 level detection

<i>CMP0</i>	
<i>GTOC_ECR_SOURCE_CMP1</i>	CMP1 level detection
<i>GTOC_ECR_SOURCE_CMP2</i>	CMP2 level detection
<i>GTOC_ECR_SOURCE_CMP3</i>	CMP3 level detection
<i>GTOC_ECR_SOURCE_GTOCPIN</i>	GTOCx_IN input level detection
<b>Input parameter{in}</b>	
<b>valid_level</b>	valid level
<i>GTOC_VALID_LEVEL_LOW</i>	low level is valid
<i>GTOC_VALID_LEVEL_HIGH</i>	high level is valid
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure extended closing request from GTOC0 */
```

```
gtoc_extended_closing_request_disable(GTOC0,          GTOC_ECR_SOURCE_CMP0,
GTOC_VALID_LEVEL_LOW);
```

### gtoc\_extended\_closing\_request\_mask

The description of gtoc\_extended\_closing\_request\_mask is shown as below:

**Table 3-580. Function gtoc\_extended\_closing\_request\_mask**

<b>Function name</b>	gtoc_extended_closing_request_mask
<b>Function prototype</b>	void gtoc_extended_closing_request_mask(uint32_t gtoc_periph, uint32_t mask_source)
<b>Function descriptions</b>	mask GTOC extended closing request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>mask_source</b>	mask source for extended closing request
<i>GTOC_ECRMKSEL_NOT_MASKED</i>	Extended closing request is not masked
<i>GTOC_ECRMKSE</i>	Extended closing request is masked by GPTIMER0_CH0

<i>L_GPTIMER0_CH0</i>	
<i>GTOC_ECRMKSE</i> <i>L_GPTIMER0_CH1</i>	Extended closing request is masked by GPTIMER0_CH1
<i>GTOC_ECRMKSE</i> <i>L_GPTIMER1_CH0</i>	Extended closing request is masked by GPTIMER1_CH0
<i>GTOC_ECRMKSE</i> <i>L_GPTIMER1_CH1</i>	Extended closing request is masked by GPTIMER1_CH1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* mask extended closing request from GTOC0 */
```

```
gtoc_extended_closing_request_mask(GTOC0, GTOC_ECRMKSEL_GPTIMER0_CH0);
```

### gtoc\_flag\_get

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

**Table 3-581. Function gtoc\_flag\_get**

<b>Function name</b>	gtoc_flag_get
<b>Function prototype</b>	FlagStatus gtoc_flag_get(uint32_t gtoc_periph, uint32_t flag)
<b>Function descriptions</b>	get GTOC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>flag</b>	GTOC flag
<i>GTOC_FLAG_INIF</i>	GTOCx_IN input interrupt flag
<i>GTOC_FLAG_OFVEIF</i>	GPTIMER output fault or CMP valid edge interrupt flag
<i>GTOC_FLAG_HXTALSD</i> <i>F</i>	HXTAL stuck detection flag
<i>GTOC_FLAG_LOCKUP</i> <i>DF</i>	CPU LOCKUP detection flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
FlagStatus status;
```

```
/* get GTOC0 flag */
```

```
status = gtoc_flag_get(GTOC0, GTOC_FLAG_INIF);
```

### gtoc\_flag\_clear

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

**Table 3-582. Function gtoc\_flag\_clear**

<b>Function name</b>	gtoc_flag_clear
<b>Function prototype</b>	void gtoc_flag_clear(uint32_t gtoc_periph, uint32_t flag)
<b>Function descriptions</b>	clear GTOC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>flag</b>	GTOC flag
<i>GTOC_FLAG_INIF</i>	GTOCx_IN input interrupt flag
<i>GTOC_FLAG_OFVEIF</i>	GPTIMER output fault or CMP valid edge interrupt flag
<i>GTOC_FLAG_HXTALSD</i> <i>F</i>	HXTAL stuck detection flag
<i>GTOC_FLAG_LOCKUP</i> <i>DF</i>	CPU LOCKUP detection flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear GTOC0 flag */
```

```
gtoc_flag_clear(GTOC0, GTOC_FLAG_INIF);
```

### gtoc\_interrupt\_flag\_get

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

**Table 3-583. Function gtoc\_interrupt\_flag\_get**

<b>Function name</b>	gtoc_interrupt_flag_get
<b>Function prototype</b>	FlagStatus gtoc_interrupt_flag_get(uint32_t gtoc_periph, uint32_t int_flag)
<b>Function descriptions</b>	get GTOC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	



<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>int_flag</b>	GTOC interrupt flag
<i>GTOC_INT_FLAG_INIF</i>	GTOCx_IN input interrupt flag
<i>GTOC_INT_FLAG_OFV EIF</i>	GPTIMER output fault or CMP valid edge interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
FlagStatus status;
```

```
/* get GTOC0 interrupt flag */
```

```
status = gtoc_interrupt_flag_get(GTOC0, GTOC_INT_FLAG_INIF);
```

### gtoc\_interrupt\_flag\_clear

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

**Table 3-584. Function gtoc\_interrupt\_flag\_clear**

<b>Function name</b>	gtoc_interrupt_flag_clear
<b>Function prototype</b>	void gtoc_interrupt_flag_clear(uint32_t gtoc_periph, uint32_t int_flag)
<b>Function descriptions</b>	clear GTOC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gtoc_periph</b>	GTOC peripheral
<i>GTOCx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>int_flag</b>	GTOC interrupt flag
<i>GTOC_INT_FLAG_INIF</i>	GTOCx_IN input interrupt flag
<i>GTOC_INT_FLAG_OFV EIF</i>	GPTIMER output fault or CMP valid edge interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear GTOC0 interrupt flag */
```

```
gtoc_interrupt_flag_clear(GTOC0, GTOC_INT_FLAG_INIF);
```

## 3.19. 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 [3.19.1](#), the I2C firmware functions are introduced in chapter [3.19.2](#).

### 3.19.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

**Table 3-585. 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

### 3.19.2. Descriptions of Peripheral functions

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

**Table 3-586. 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_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

Function name	Function description
i2c_address10_disable	disable 10-bit addressing mode in master mode
i2c_automatic_end_enable	enable I2C automatic end mode in master mode
i2c_automatic_end_disable	disable I2C automatic end mode in master mode
i2c_slave_response_to_gcall_enable	enable the response to a general call
i2c_slave_response_to_gcall_disable	disable the response to a general call
i2c_stretch_scl_low_enable	enable to stretch SCL low when data is not ready in slave mode
i2c_stretch_scl_low_disable	disable to stretch SCL low when data is not ready in slave mode
i2c_address_config	configure i2c slave address
i2c_address_bit_compare_config	define which bits of ADDRESS[7:1] need to compare with the incoming address byte
i2c_address_disable	disable I2C address in slave mode
i2c_second_address_config	configure I2C second slave address
i2c_second_address_disable	disable I2C second address in slave mode
i2c_receved_address_get	get received match address in slave mode
i2c_slave_byte_control_enable	enable slave byte control
i2c_slave_byte_control_disable	disable slave byte control
i2c_nack_enable	generate a NACK in slave mode
i2c_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_mode_enable	enable SMBus mode
i2c_smbus_mode_disable	disable SMBus mode
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

Function name	Function description
i2c_smbus_host_addr_enable	enable SMBus host address
i2c_smbus_host_addr_disable	disable SMBus host address
i2c_extented_clock_timeout_enable	enable extended clock timeout detection
i2c_extented_clock_timeout_disable	disable extended clock timeout detection
i2c_clock_timeout_enable	enable clock timeout detection
i2c_clock_timeout_disable	disable clock timeout detection
i2c_bus_timeout_b_config	configure bus timeout B
i2c_bus_timeout_a_config	configure bus timeout A
i2c_idle_clock_timeout_config	configure idle clock timeout detection
i2c_flag_get	get I2C flag status
i2c_flag_clear	clear I2C flag status
i2c_interrupt_enable	enable I2C interrupt
i2c_interrupt_disable	disable I2C interrupt
i2c_interrupt_flag_get	get I2C interrupt flag status
i2c_interrupt_flag_clear	clear I2C interrupt flag status

### Enum i2c\_interrupt\_flag\_enum

Table 3-587. 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 3-588. Function i2c\_deinit

Function name	i2c_deinit
Function prototype	void i2c_deinit(void);
Function descriptions	reset I2C
Precondition	-

<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 I2C */
```

```
i2c_deinit();
```

### i2c\_timing\_config

The description of i2c\_timing\_config is shown as below:

**Table 3-589. Function i2c\_timing\_config**

<b>Function name</b>	i2c_timing_config
<b>Function prototype</b>	void i2c_timing_config(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>	-
<b>Input parameter{in}</b>	
<b>psc</b>	0-0xf, timing prescaler
<b>Input parameter{in}</b>	
<b>scl_dely</b>	0-0xf,data setup time
<b>Input parameter{in}</b>	
<b>sda_dely</b>	0-0xf,data hold time
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the timing parameters */
```

```
i2c_timing_config(0x1, 0x2, 0x1);
```

### i2c\_digital\_noise\_filter\_config

The description of i2c\_digital\_noise\_filter\_config is shown as below:

**Table 3-590. 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 filter_length);

<b>Function descriptions</b>	configure digital noise filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<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 $t_{I2CCLK}$
<i>FILTER_LENGTH_2</i>	digital filter is enabled and filter spikes with a length of up to 2 $t_{I2CCLK}$
<i>FILTER_LENGTH_3</i>	digital filter is enabled and filter spikes with a length of up to 3 $t_{I2CCLK}$
<i>FILTER_LENGTH_4</i>	digital filter is enabled and filter spikes with a length of up to 4 $t_{I2CCLK}$
<i>FILTER_LENGTH_5</i>	digital filter is enabled and filter spikes with a length of up to 5 $t_{I2CCLK}$
<i>FILTER_LENGTH_6</i>	digital filter is enabled and filter spikes with a length of up to 6 $t_{I2CCLK}$
<i>FILTER_LENGTH_7</i>	digital filter is enabled and filter spikes with a length of up to 7 $t_{I2CCLK}$
<i>FILTER_LENGTH_8</i>	digital filter is enabled and filter spikes with a length of up to 8 $t_{I2CCLK}$
<i>FILTER_LENGTH_9</i>	digital filter is enabled and filter spikes with a length of up to 9 $t_{I2CCLK}$
<i>FILTER_LENGTH_10</i>	digital filter is enabled and filter spikes with a length of up to 10 $t_{I2CCLK}$
<i>FILTER_LENGTH_11</i>	digital filter is enabled and filter spikes with a length of up to 11 $t_{I2CCLK}$
<i>FILTER_LENGTH_12</i>	digital filter is enabled and filter spikes with a length of up to 12 $t_{I2CCLK}$
<i>FILTER_LENGTH_13</i>	digital filter is enabled and filter spikes with a length of up to 13 $t_{I2CCLK}$
<i>FILTER_LENGTH_14</i>	digital filter is enabled and filter spikes with a length of up to 14 $t_{I2CCLK}$
<i>FILTER_LENGTH_15</i>	digital filter is enabled and filter spikes with a length of up to 15 $t_{I2CCLK}$
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* I2C digital filter filters spikes with a length of up to 1  $t_{I2CCLK}$  */
```

```
i2c_digital_noise_filter_config(FILTER_LENGTH_1);
```

## i2c\_master\_clock\_config

The description of i2c\_master\_clock\_config is shown as below:

**Table 3-591. Function i2c\_master\_clock\_config**

<b>Function name</b>	i2c_master_clock_config
<b>Function prototype</b>	void i2c_master_clock_config(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>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(0x0f, 0x0f);
```

### i2c\_master\_addressing

The description of i2c\_master\_addressing is shown as below:

**Table 3-592. Function i2c\_master\_addressing**

<b>Function name</b>	i2c_master_addressing
<b>Function prototype</b>	void i2c_master_addressing(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>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(0x82, I2C_MASTER_TRANSMIT);
```

### i2c\_address10\_header\_enable

The description of i2c\_address10\_header\_enable is shown as below:

**Table 3-593. Function i2c\_address10\_header\_enable**

<b>Function name</b>	i2c_address10_header_enable
<b>Function prototype</b>	void i2c_address10_header_enable(void);

<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>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();
```

### i2c\_address10\_header\_disable

The description of i2c\_address10\_header\_disable is shown as below:

**Table 3-594. Function i2c\_address10\_header\_disable**

<b>Function name</b>	i2c_address10_header_disable
<b>Function prototype</b>	void i2c_address10_header_disable(void);
<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>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* 10-bit address header executes complete sequence in master receive mode */
```

```
i2c_address10_header_disable();
```

### i2c\_address10\_enable

The description of i2c\_address10\_enable is shown as below:

**Table 3-595. Function i2c\_address10\_enable**

<b>Function name</b>	i2c_address10_enable
<b>Function prototype</b>	void i2c_address10_enable(void);
<b>Function descriptions</b>	enable 10-bit addressing mode in master mode
<b>Precondition</b>	-



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

Example:

```
/* enable 10-bit addressing mode in master mode */
```

```
i2c_address10_enable();
```

### i2c\_address10\_disable

The description of i2c\_address10\_disable is shown as below:

**Table 3-596. Function i2c\_address10\_disable**

Function name	i2c_address10_disable
Function prototype	void i2c_address10_disable(void);
Function descriptions	disable 10-bit addressing mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable 10-bit addressing mode in master mode */
```

```
i2c_address10_disable();
```

### i2c\_automatic\_end\_enable

The description of i2c\_automatic\_end\_enable is shown as below:

**Table 3-597. Function i2c\_automatic\_end\_enable**

Function name	i2c_automatic_end_enable
Function prototype	void i2c_automatic_end_enable(void);
Function descriptions	enable I2C automatic end mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* enable I2C automatic end mode in master mode */
```

```
i2c_automatic_end_enable();
```

### i2c\_automatic\_end\_disable

The description of i2c\_automatic\_end\_disable is shown as below:

**Table 3-598. Function i2c\_automatic\_end\_disable**

<b>Function name</b>	i2c_automatic_end_disable
<b>Function prototype</b>	void i2c_automatic_end_disable(void);
<b>Function descriptions</b>	disable I2C automatic end mode in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C automatic end mode in master mode */
```

```
i2c_automatic_end_disable();
```

### i2c\_slave\_response\_to\_gcall\_enable

The description of i2c\_slave\_response\_to\_gcall\_enable is shown as below:

**Table 3-599. 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(void);
<b>Function descriptions</b>	enable the response to a general call
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* enable the response to a general call */
```

```
i2c_slave_response_to_gcall_enable();
```

### i2c\_slave\_response\_to\_gcall\_disable

The description of i2c\_slave\_response\_to\_gcall\_disable is shown as below:

**Table 3-600. Function i2c\_slave\_response\_to\_gcall\_disable**

<b>Function name</b>	i2c_slave_response_to_gcall_disable
<b>Function prototype</b>	void i2c_slave_response_to_gcall_disable(void);
<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>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the response to a general call */
```

```
i2c_slave_response_to_gcall_disable();
```

### i2c\_stretch\_scl\_low\_enable

The description of i2c\_stretch\_scl\_low\_enable is shown as below:

**Table 3-601. Function i2c\_stretch\_scl\_low\_enable**

<b>Function name</b>	i2c_stretch_scl_low_enable
<b>Function prototype</b>	void i2c_stretch_scl_low_enable(void);
<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>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();
```

### **i2c\_stretch\_scl\_low\_disable**

The description of i2c\_stretch\_scl\_low\_disable is shown as below:

**Table 3-602. 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(void);
<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>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();
```

### **i2c\_address\_config**

The description of i2c\_address\_config is shown as below:

**Table 3-603. Function i2c\_address\_config**

<b>Function name</b>	i2c_address_config
<b>Function prototype</b>	void i2c_address_config(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>address</b>	I2C address
<b>Input parameter{in}</b>	
<b>addr_format</b>	7 bits or 10 bits
I2C_ADDFORMAT_7BITS	7bits
I2C_ADDFORMAT_10	10bits

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

Example:

```
/* configure i2c slave address */
```

```
i2c_address_config(0x82, I2C_ADDFORMAT_7BITS);
```

### **i2c\_address\_bit\_compare\_config**

The description of i2c\_address\_bit\_compare\_config is shown as below:

**Table 3-604. Function i2c\_address\_bit\_compare\_config**

<b>Function name</b>	i2c_address_bit_compare_config
<b>Function prototype</b>	void i2c_address_bit_compare_config(uint32_t compare_bits);
<b>Function descriptions</b>	define which bits of ADDRESS[7:1] need to compare with the incoming address byte
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>compare_bits</b>	the bits need to compare
ADDRESS_NO_COMPARE	all the ADDRESS[7:1] bits are masked
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
ADDRESS_ALL_COMPARE	no mask, all these bits must be compared
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* bit 1 of ADDRESS[7:1] need to compare with the incoming address byte */
```

```
i2c_address_bit_compare_config(ADDRESS_BIT1_COMPARE);
```

### i2c\_address\_disable

The description of i2c\_address\_disable is shown as below:

**Table 3-605. Function i2c\_address\_disable**

<b>Function name</b>	i2c_address_disable
<b>Function prototype</b>	void i2c_address_disable(void);
<b>Function descriptions</b>	disable i2c address in slave 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 i2c address in slave mode */
```

```
i2c_address_disable();
```

### i2c\_second\_address\_config

The description of i2c\_second\_address\_config is shown as below:

**Table 3-606. Function i2c\_second\_address\_config**

<b>Function name</b>	i2c_second_address_config
<b>Function prototype</b>	void i2c_second_address_config(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>address</b>	I2C address
<b>Input parameter{in}</b>	
<b>addr_mask</b>	the bits not need to compare
ADDRESS2_NO_MASK	no mask, all the bits must be compared
ADDRESS2_MASK_BIT	ADDRESS2[1] is masked, only ADDRESS2[7:2] are compared

<i>T1</i>	
<i>ADDRESS2_MASK_BIT1_2</i>	ADDRESS2[2:1] is masked, only ADDRESS2[7:3] are compared
<i>ADDRESS2_MASK_BIT1_3</i>	ADDRESS2[3:1] is masked, only ADDRESS2[7:4] are compared
<i>ADDRESS2_MASK_BIT1_4</i>	ADDRESS2[4:1] is masked, only ADDRESS2[7:5] are compared
<i>ADDRESS2_MASK_BIT1_5</i>	ADDRESS2[5:1] is masked, only ADDRESS2[7:6] are compared
<i>ADDRESS2_MASK_BIT1_6</i>	ADDRESS2[6:1] is masked, only ADDRESS2[7] are compared
<i>ADDRESS2_MASK_ALL</i>	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(0x82, ADDRESS2_MASK_BIT1_2);
```

### i2c\_second\_address\_disable

The description of i2c\_second\_address\_disable is shown as below:

**Table 3-607. Function i2c\_second\_address\_disable**

<b>Function name</b>	i2c_second_address_disable
<b>Function prototype</b>	void i2c_second_address_disable(void);
<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>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable i2c second address in slave mode */
```

```
i2c_second_address_disable();
```

## i2c\_receved\_address\_get

The description of i2c\_receved\_address\_get is shown as below:

**Table 3-608. Function i2c\_receved\_address\_get**

<b>Function name</b>	i2c_receved_address_get
<b>Function prototype</b>	uint32_t i2c_receved_address_get(void);
<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>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	0x00..0x7F

Example:

```
/* get received match address in slave mode */
uint32_t address;

address = i2c_receved_address_get();
```

## i2c\_slave\_byte\_control\_enable

The description of i2c\_slave\_byte\_control\_enable is shown as below:

**Table 3-609. 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(void);
<b>Function descriptions</b>	enable slave byte control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable slave byte control */

i2c_slave_byte_control_enable();
```



## i2c\_slave\_byte\_control\_disable

The description of i2c\_slave\_byte\_control\_disable is shown as below:

**Table 3-610. 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(void);
<b>Function descriptions</b>	disable slave byte control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable slave byte control */
i2c_slave_byte_control_disable();
```

## i2c\_nack\_enable

The description of i2c\_nack\_enable is shown as below:

**Table 3-611. Function i2c\_nack\_enable**

<b>Function name</b>	i2c_nack_enable
<b>Function prototype</b>	void i2c_nack_enable(void);
<b>Function descriptions</b>	generate a NACK in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* generate a NACK in slave mode */
i2c_nack_enable();
```

## i2c\_wakeup\_from\_deepsleep\_enable

The description of i2c\_wakeup\_from\_deepsleep\_enable is shown as below:

**Table 3-612. 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(void);
<b>Function descriptions</b>	enable wakeup from Deep-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:

```
/* enable wakeup from Deep-sleep mode */
i2c_wakeup_from_deepsleep_enable();
```

### i2c\_wakeup\_from\_deepsleep\_disable

The description of i2c\_wakeup\_from\_deepsleep\_disable is shown as below:

**Table 3-613. 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(void);
<b>Function descriptions</b>	disable wakeup from Deep-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:

```
/* disable wakeup from Deep-sleep mode */
i2c_wakeup_from_deepsleep_disable();
```

### i2c\_enable

The description of i2c\_enable is shown as below:

**Table 3-614. Function i2c\_enable**

<b>Function name</b>	i2c_enable
----------------------	------------

<b>Function prototype</b>	void i2c_enable(void);
<b>Function descriptions</b>	enable I2C
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I2C */
```

```
i2c_enable();
```

### i2c\_disable

The description of i2c\_disable is shown as below:

**Table 3-615. Function i2c\_disable**

<b>Function name</b>	i2c_disable
<b>Function prototype</b>	void i2c_disable(void);
<b>Function descriptions</b>	disable I2C
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable I2C */
```

```
i2c_disable();
```

### i2c\_start\_on\_bus

The description of i2c\_start\_on\_bus is shown as below:

**Table 3-616. Function i2c\_start\_on\_bus**

<b>Function name</b>	i2c_start_on_bus
<b>Function prototype</b>	void i2c_start_on_bus(void);
<b>Function descriptions</b>	generate a START condition on I2C bus

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

Example:

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

```
i2c_start_on_bus();
```

### i2c\_stop\_on\_bus

The description of i2c\_stop\_on\_bus is shown as below:

**Table 3-617. Function i2c\_stop\_on\_bus**

Function name	i2c_stop_on_bus
Function prototype	void i2c_stop_on_bus(void);
Function descriptions	generate a STOP condition on I2C bus
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* generate a STOP condition to I2C bus */
```

```
i2c_stop_on_bus();
```

### i2c\_data\_transmit

The description of i2c\_data\_transmit is shown as below:

**Table 3-618. Function i2c\_data\_transmit**

Function name	i2c_data_transmit
Function prototype	void i2c_data_transmit(uint8_t data);
Function descriptions	I2C transmit data
Precondition	-
The called functions	-

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

Example:

```
/* I2C transmit data */
i2c_data_transmit(0x80);
```

### i2c\_data\_receive

The description of i2c\_data\_receive is shown as below:

**Table 3-619. Function i2c\_data\_receive**

Function name	i2c_data_receive
Function prototype	uint8_t i2c_data_receive(void);
Function descriptions	I2C receive data
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	received data (0x00-0xFF)

Example:

```
/* I2C receive data */
uint32_t i2c_receiver;
i2c_receiver = i2c_data_receive();
```

### i2c\_reload\_enable

The description of i2c\_reload\_enable is shown as below:

**Table 3-620. Function i2c\_reload\_enable**

Function name	i2c_reload_enable
Function prototype	void i2c_reload_enable(void);
Function descriptions	enable I2C reload mode
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* enable I2C reload mode */
```

```
i2c_reload_enable();
```

### i2c\_reload\_disable

The description of i2c\_reload\_disable is shown as below:

**Table 3-621. Function i2c\_reload\_disable**

<b>Function name</b>	i2c_reload_disable
<b>Function prototype</b>	void i2c_reload_disable(void);
<b>Function descriptions</b>	disable I2C reload mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C reload mode */
```

```
i2c_reload_disable();
```

### i2c\_transfer\_byte\_number\_config

The description of i2c\_transfer\_byte\_number\_config is shown as below:

**Table 3-622. Function i2c\_transfer\_byte\_number\_config**

<b>Function name</b>	i2c_transfer_byte_number_config
<b>Function prototype</b>	void i2c_transfer_byte_number_config(uint8_t byte_number);
<b>Function descriptions</b>	configure number of bytes to be transferred
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>byte_number</b>	0x0-0xFF, number of bytes to be transferred
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure number of bytes to be transferred */
i2c_transfer_byte_number_config(0xFF);
```

### i2c\_dma\_enable

The description of i2c\_dma\_enable is shown as below:

**Table 3-623. Function i2c\_dma\_enable**

<b>Function name</b>	i2c_dma_enable
<b>Function prototype</b>	void i2c_dma_enable(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>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(I2C_DMA_RECEIVE);
```

### i2c\_dma\_disable

The description of i2c\_dma\_disable is shown as below:

**Table 3-624. Function i2c\_dma\_disable**

<b>Function name</b>	i2c_dma_disable
<b>Function prototype</b>	void i2c_dma_disable(uint8_t dma);
<b>Function descriptions</b>	disable I2C DMA for transmission or reception
<b>Precondition</b>	-
<b>The called functions</b>	-
<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:

```
/* disable I2C DMA for transmission or reception */
```

```
i2c_dma_disable(I2C_DMA_RECEIVE);
```

### **i2c\_pec\_transfer**

The description of i2c\_pec\_transfer is shown as below:

**Table 3-625. Function i2c\_pec\_transfer**

<b>Function name</b>	i2c_pec_transfer
<b>Function prototype</b>	void i2c_pec_transfer(void);
<b>Function descriptions</b>	I2C transfers PEC value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* I2C transfers PEC value */
```

```
i2c_pec_transfer();
```

### **i2c\_pec\_enable**

The description of i2c\_pec\_enable is shown as below:

**Table 3-626. Function i2c\_pec\_enable**

<b>Function name</b>	i2c_pec_enable
<b>Function prototype</b>	void i2c_pec_enable(void);
<b>Function descriptions</b>	enable I2C PEC calculation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	



-	-
Return value	
-	-

Example:

```
/* enable I2C PEC calculation */
```

```
i2c_pec_enable();
```

### i2c\_pec\_disable

The description of i2c\_pec\_disable is shown as below:

**Table 3-627. Function i2c\_pec\_disable**

<b>Function name</b>	i2c_pec_disable
<b>Function prototype</b>	void i2c_pec_disable(void);
<b>Function descriptions</b>	disable I2C PEC calculation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable I2C PEC calculation */
```

```
i2c_pec_disable();
```

### i2c\_pec\_value\_get

The description of i2c\_pec\_value\_get is shown as below:

**Table 3-628. Function i2c\_pec\_value\_get**

<b>Function name</b>	i2c_pec_value_get
<b>Function prototype</b>	uint32_t i2c_pec_value_get(void);
<b>Function descriptions</b>	get packet error checking 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	PEC value
----------	-----------

Example:

```
/* get packet error checking value */

uint32_t pec_value;

pec_value = i2c_pec_value_get();
```

### i2c\_smbus\_mode\_enable

The description of i2c\_smbus\_mode\_enable is shown as below:

**Table 3-629. Function i2c\_smbus\_mode\_enable**

<b>Function name</b>	i2c_smbus_mode_enable
<b>Function prototype</b>	void i2c_smbus_mode_enable(void);
<b>Function descriptions</b>	enable SMBus 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 SMBus mode */

i2c_smbus_mode_enable();
```

### i2c\_smbus\_mode\_disable

The description of i2c\_smbus\_mode\_disable is shown as below:

**Table 3-630. Function i2c\_smbus\_mode\_disable**

<b>Function name</b>	i2c_smbus_mode_disable
<b>Function prototype</b>	void i2c_smbus_mode_disable(void);
<b>Function descriptions</b>	disable SMBus 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 SMBus mode */

i2c_smbus_mode_disable();
```

### i2c\_smbus\_alert\_enable

The description of i2c\_smbus\_alert\_enable is shown as below:

**Table 3-631. Function i2c\_smbus\_alert\_enable**

<b>Function name</b>	i2c_smbus_alert_enable
<b>Function prototype</b>	void i2c_smbus_alert_enable(void);
<b>Function descriptions</b>	enable SMBus Alert
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SMBus Alert */

i2c_smbus_alert_enable();
```

### i2c\_smbus\_alert\_disable

The description of i2c\_smbus\_alert\_disable is shown as below:

**Table 3-632. Function i2c\_smbus\_alert\_disable**

<b>Function name</b>	i2c_smbus_alert_disable
<b>Function prototype</b>	void i2c_smbus_alert_disable(void);
<b>Function descriptions</b>	disable SMBus Alert
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SMBus Alert */
```

```
i2c_smbus_alert_disable();
```

### i2c\_smbus\_default\_addr\_enable

The description of i2c\_smbus\_default\_addr\_enable is shown as below:

**Table 3-633. 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(void);
<b>Function descriptions</b>	enable SMBus device default address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SMBus device default address */
```

```
i2c_smbus_default_addr_enable();
```

### i2c\_smbus\_default\_addr\_disable

The description of i2c\_smbus\_default\_addr\_disable is shown as below:

**Table 3-634. 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(void);
<b>Function descriptions</b>	disable SMBus device default address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SMBus device default address */
```

```
i2c_smbus_default_addr_disable();
```

## i2c\_smbus\_host\_addr\_enable

The description of i2c\_smbus\_host\_addr\_enable is shown as below:

**Table 3-635. 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(void);
<b>Function descriptions</b>	enable SMBus Host address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SMBus Host address */
i2c_smbus_host_addr_enable();
```

## i2c\_smbus\_host\_addr\_disable

The description of i2c\_smbus\_host\_addr\_disable is shown as below:

**Table 3-636. 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(void);
<b>Function descriptions</b>	disable SMBus Host address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SMBus Host address */
i2c_smbus_host_addr_disable();
```

## i2c\_extented\_clock\_timeout\_enable

The description of i2c\_extented\_clock\_timeout\_enable is shown as below:

**Table 3-637. 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(void);
<b>Function descriptions</b>	enable extended clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable extended clock timeout detection */
i2c_extented_clock_timeout_enable();
```

### **i2c\_extented\_clock\_timeout\_disable**

The description of i2c\_extented\_clock\_timeout\_disable is shown as below:

**Table 3-638. 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(void);
<b>Function descriptions</b>	disable extended clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable extended clock timeout detection */
i2c_extented_clock_timeout_disable();
```

### **i2c\_clock\_timeout\_enable**

The description of i2c\_clock\_timeout\_enable is shown as below:

**Table 3-639. Function i2c\_clock\_timeout\_enable**

<b>Function name</b>	i2c_clock_timeout_enable
----------------------	--------------------------

<b>Function prototype</b>	void i2c_clock_timeout_enable(void);
<b>Function descriptions</b>	enable clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable clock timeout detection */
i2c_clock_timeout_enable();
```

### i2c\_clock\_timeout\_disable

The description of i2c\_clock\_timeout\_disable is shown as below:

**Table 3-640. Function i2c\_clock\_timeout\_disable**

<b>Function name</b>	i2c_clock_timeout_disable
<b>Function prototype</b>	void i2c_clock_timeout_disable(void);
<b>Function descriptions</b>	disable clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable clock timeout detection */
i2c_clock_timeout_disable();
```

### i2c\_bus\_timeout\_b\_config

The description of i2c\_bus\_timeout\_b\_config is shown as below:

**Table 3-641. 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 timeout);
<b>Function descriptions</b>	configure bus timeout B

<b>Precondition</b>	-
<b>The called functions</b>	-
<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(0xff);
```

### **i2c\_bus\_timeout\_a\_config**

The description of i2c\_bus\_timeout\_a\_config is shown as below:

**Table 3-642. 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 timeout);
<b>Function descriptions</b>	configure bus timeout A
<b>Precondition</b>	-
<b>The called functions</b>	-
<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(0xff);
```

### **i2c\_idle\_clock\_timeout\_config**

The description of i2c\_idle\_clock\_timeout\_config is shown as below:

**Table 3-643. 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 timeout);
<b>Function descriptions</b>	configure idle clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-



Input parameter{in}	
<b>timeout</b>	bus timeout A
<i>BUSTOA_DETECT_SCL_LOW</i>	BUSTOA is used to detect SCL low timeout
<i>BUSTOA_DETECT_IDLE</i>	BUSTOA is used to detect both SCL and SDA high timeout when the bus is idle
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure idle clock timeout detection */
```

```
i2c_idle_clock_timeout_config(BUSTOA_DETECT_SCL_LOW);
```

### i2c\_flag\_get

The description of i2c\_flag\_get is shown as below:

**Table 3-644. Function i2c\_flag\_get**

<b>Function name</b>	i2c_flag_get
<b>Function prototype</b>	FlagStatus i2c_flag_get(uint32_t flag);
<b>Function descriptions</b>	get I2C flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<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}	

-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET / RESET

Example:

```
/* get I2C flag status */
```

```
FlagStatus flag_state = RESET;
```

```
flag_state = i2c_flag_get(I2C_FLAG_TBE);
```

### i2c\_flag\_clear

The description of i2c\_flag\_clear is shown as below:

**Table 3-645. Function i2c\_flag\_clear**

<b>Function name</b>	i2c_flag_clear
<b>Function prototype</b>	void i2c_flag_clear(uint32_t flag);
<b>Function descriptions</b>	clear I2C flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	I2C flags
<i>I2C_FLAG_ADDSEND</i>	address received matches in slave mode
<i>I2C_FLAG_NACK</i>	not acknowledge flag
<i>I2C_FLAG_STPDET</i>	STOP condition detected in slave mode
<i>I2C_FLAG_BERR</i>	bus error
<i>I2C_FLAG_LOSTARB</i>	arbitration Lost
<i>I2C_FLAG_OUERR</i>	overflow/underrun error in slave mode
<i>I2C_FLAG_PECERR</i>	PEC error
<i>I2C_FLAG_TIMEOUT</i>	timeout flag
<i>I2C_FLAG_SMBALT</i>	SMBus Alert
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear a bus error flag*/
```

```
i2c_flag_clear(I2C_FLAG_BERR);
```

### i2c\_interrupt\_enable

The description of i2c\_interrupt\_enable is shown as below:

Table 3-646. Function `i2c_interrupt_enable`

<b>Function name</b>	<code>i2c_interrupt_enable</code>
<b>Function prototype</b>	<code>void i2c_interrupt_enable(uint32_t interrupt);</code>
<b>Function descriptions</b>	enable I2C interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	I2C interrupts
<code>I2C_INT_ERR</code>	error interrupt
<code>I2C_INT_TC</code>	transfer complete interrupt
<code>I2C_INT_STPDET</code>	stop detection interrupt
<code>I2C_INT_NACK</code>	not acknowledge received interrupt
<code>I2C_INT_ADDM</code>	address match interrupt
<code>I2C_INT_RBNE</code>	receive interrupt
<code>I2C_INT_TI</code>	transmit interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I2C transmit interrupt */
i2c_interrupt_enable(I2C_INT_TI);
```

### `i2c_interrupt_disable`

The description of `i2c_interrupt_disable` is shown as below:

Table 3-647. Function `i2c_interrupt_disable`

<b>Function name</b>	<code>i2c_interrupt_disable</code>
<b>Function prototype</b>	<code>void i2c_interrupt_disable(uint32_t interrupt);</code>
<b>Function descriptions</b>	disable I2C interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	I2C interrupts
<code>I2C_INT_ERR</code>	error interrupt
<code>I2C_INT_TC</code>	transfer complete interrupt
<code>I2C_INT_STPDET</code>	stop detection interrupt
<code>I2C_INT_NACK</code>	not acknowledge received interrupt
<code>I2C_INT_ADDM</code>	address match interrupt
<code>I2C_INT_RBNE</code>	receive interrupt
<code>I2C_INT_TI</code>	transmit interrupt

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C transmit interrupt */
```

```
i2c_interrupt_disable(I2C_INT_TI);
```

### i2c\_interrupt\_flag\_get

The description of i2c\_interrupt\_flag\_get is shown as below:

**Table 3-648. Function i2c\_interrupt\_flag\_get**

Function name	i2c_interrupt_flag_get
Function prototype	FlagStatus i2c_interrupt_flag_get(i2c_interrupt_flag_enum int_flag);
Function descriptions	get I2C interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	I2C interrupt flags, refer to <a href="#">Table 3-587. Enum i2c_interrupt_flag_enum.</a>
I2C_INT_FLAG_TI	transmit interrupt flag
I2C_INT_FLAG_RBNE	I2C_RDATA is not empty during receiving interrupt flag
I2C_INT_FLAG_ADDS END	address received matches in slave mode interrupt flag
I2C_INT_FLAG_NACK	not acknowledge interrupt flag
I2C_INT_FLAG_STPD ET	stop condition detected in slave mode interrupt flag
I2C_INT_FLAG_TC	transfer complete in master mode interrupt flag
I2C_INT_FLAG_TCR	transfer complete reload interrupt flag
I2C_INT_FLAG_BERR	bus error interrupt flag
I2C_INT_FLAG_LOSTA RB	arbitration lost interrupt flag
I2C_INT_FLAG_OUER R	overflow/underrun error in slave mode interrupt flag
I2C_INT_FLAG_PECE RR	PEC error interrupt flag
I2C_INT_FLAG_TIMEO UT	timeout interrupt flag
I2C_INT_FLAG_SMBA LT	SMBus Alert interrupt flag
Output parameter{out}	
-	-
Return value	

<b>FlagStatus</b>	SET / RESET
-------------------	-------------

Example:

```
/* get I2C interrupt flag status */
```

```
FlagStatus flag_state = RESET;
```

```
flag_state = i2c_interrupt_flag_get(I2C_INT_FLAG_TI);
```

### i2c\_interrupt\_flag\_clear

The description of i2c\_interrupt\_flag\_clear is shown as below:

**Table 3-649. Function i2c\_interrupt\_flag\_clear**

<b>Function name</b>	i2c_interrupt_flag_clear
<b>Function prototype</b>	void i2c_interrupt_flag_clear(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>int_flag</b>	I2C interrupt flags, refer to <a href="#">Table 3-587. Enum i2c_interrupt_flag_enum</a> .
<i>I2C_INT_FLAG_ADDS END</i>	address received matches in slave mode interrupt flag
<i>I2C_INT_FLAG_NACK</i>	not acknowledge interrupt flag
<i>I2C_INT_FLAG_STPD ET</i>	stop condition detected in slave mode interrupt flag
<i>I2C_INT_FLAG_BERR</i>	bus error interrupt flag
<i>I2C_INT_FLAG_LOSTA RB</i>	arbitration lost interrupt flag
<i>I2C_INT_FLAG_OUER R</i>	overflow/underrun error in slave mode interrupt flag
<i>I2C_INT_FLAG_PECE RR</i>	PEC error interrupt flag
<i>I2C_INT_FLAG_TIMEO UT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBA LT</i>	SMBus Alert interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear a bus error flag */
```

```
i2c_interrupt_flag_clear(I2C_INT_FLAG_BERR);
```

## 3.20. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick. The NVIC and SysTick registers are listed in chapter [3.20.1](#), the MISC firmware functions are introduced in chapter [3.20.2](#).

### 3.20.1. Descriptions of Peripheral registers

**Table 3-650. NVIC Registers**

Registers	Descriptions
ISER <sup>(1)</sup>	Interrupt Set Enable Register
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
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

1. refer to the structure NVIC\_Type, is defined in the core\_cm33.h file

2. refer to the structure SCB\_Type, is defined in the core\_cm33h file

**Table 3-651. 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

### 3.20.2. Descriptions of Peripheral functions

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

**Table 3-652. MISC firmware function**

Function name	Function description
nvic_priority_group_set	set the priority group

Function name	Function description
<code>nvic_irq_enable</code>	enable NVIC interrupt request
<code>nvic_irq_disable</code>	disable NVIC interrupt request
<code>nvic_system_reset</code>	initiates a system reset request to reset the MCU
<code>nvic_vector_table_set</code>	set the NVIC vector table base address
<code>system_lowpower_set</code>	set the state of the low power mode
<code>system_lowpower_reset</code>	reset the state of the low power mode
<code>systick_clksource_set</code>	set the systick clock source

## Enum IRQn\_Type

**Table 3-653. IRQn\_Type**

Member name	Function description
<code>WWDGT_IRQn</code>	window watchDog timer interrupt
<code>LVD0_IRQn</code>	LVD0 through EXTI line 16 detect interrupt
<code>LVD1_IRQn</code>	LVD1 through EXTI line 17 detect interrupt
<code>FMC_IRQn</code>	FMC global interrupt
<code>RCU_IRQn</code>	RCU global interrupt
<code>EXTI0_IRQn</code>	EXTI line 0 interrupt
<code>EXTI1_IRQn</code>	EXTI line 1 interrupt
<code>EXTI2_IRQn</code>	EXTI line 2 interrupt
<code>EXTI3_IRQn</code>	EXTI line 3 interrupt
<code>EXTI4_IRQn</code>	EXTI line 4 interrupt
<code>DMA0_Channel0_IRQn</code>	DMA0 channel 0 global interrupt
<code>DMA0_Channel1_IRQn</code>	DMA0 channel 1 global interrupt
<code>DMA0_Channel2_IRQn</code>	DMA0 channel 2 global interrupt
<code>DMA0_Channel3_IRQn</code>	DMA0 channel 3 global interrupt
<code>DMA0_Channel4_IRQn</code>	DMA0 channel 4 global interrupt
<code>DMA0_Channel5_IRQn</code>	DMA0 channel 5 global interrupt
<code>ADC0_IRQn</code>	ADC0 interrupt
<code>CAN_TX_IRQn</code>	CAN TX interrupt
<code>CAN_RX0_IRQn</code>	CAN RX0 interrupt
<code>CAN_RX1_IRQn</code>	CAN RX1 interrupt
<code>CAN_EWMC_IRQn</code>	CAN EWMC interrupt
<code>EXTI5_9_IRQn</code>	EXTI[9:5] interrupts
<code>TIMER0_BRK_IRQn</code>	TIMER0 break interrupt
<code>TIMER0_UP_IRQn</code>	TIMER0 update interrupt
<code>TIMER0_TRG_CMT_IRQn</code>	TIMER0 trigger and channel commutation interrupt
<code>TIMER0_Channel_IRQn</code>	TIMER0 channel capture compare interrupt
<code>TIMER1_IRQn</code>	TIMER1 interrupt
<code>TIMER2_IRQn</code>	TIMER2 interrupt
<code>GPTIMER0_IRQn</code>	GPTIMER0 interrupt
<code>I2C_EV_IRQn</code>	I2C event interrupt

Member name	Function description
I2C_ER_IRQn	I2C error interrupt
SPI_IRQn	SPI global interrupt
UART0_IRQn	UART0 global interrupt
UART1_IRQn	UART1 global interrupt
EXTI10_15_IRQn	EXTI[15:10] interrupts
TIMER7_BRK_IRQn	TIMER7 Break interrupt
TIMER7_UP_IRQn	TIMER7 update interrupt
TIMER7_TRG_CMT_IRQn	TIMER7 trigger and commutation interrupt
TIMER7_Channel_IRQn	TIMER7 channel capture compare interrupt
TMU_IRQn	TMU interrupt
GPTIMER1_IRQn	GPTIMER1 interrupt
UART2_IRQn	UART2 interrupt
UART3_IRQn	UART3 interrupt
CPTIMER0_IRQn	CPTIMER0 global interrupt
CPTIMER1_IRQn	CPTIMER1 global interrupt
DMA1_Channel0_IRQn	DMA1 Channel 0 global interrupt
DMA1_Channel1_IRQn	DMA1 Channel 1 global interrupt
DMA1_Channel2_IRQn	DMA1 Channel 2 global interrupt
DMA1_Channel3_IRQn	DMA1 Channel 3 global interrupt
DMA1_Channel4_IRQn	DMA1 Channel 4 global interrupt
DMA1_Channel5_IRQn	DMA1 Channel 5 global interrupt
DMAMUX_OVERRUN_IRQn	DMAMUX overrun interrupt
CPTIMERW_IRQn	CPTIMERW global interrupt
CFMU_IRQn	CFMU interrupt
I2C_WKUP_IRQn	I2C wakeup through EXTI line 23 detect interrupt
FWDGT_IRQn	FWDGT through EXTI line 22 detect interrupt
CMP0_IRQn	CMP0 interrupt
CMP1_IRQn	CMP1 interrupt
CMP2_IRQn	CMP2 interrupt
CMP3_IRQn	CMP3 interrupt
ADC2_IRQn	ADC2 interrupt
EVIC_IRQn	EVIC global interrupt
GTOC0_IRQn	GTOC0 interrupt
GTOC1_IRQn	GTOC1 interrupt
GTOC2_IRQn	GTOC2 interrupt
GTOC3_IRQn	GTOC3 interrupt
CMP0_EXTI_IRQn	CMP0 through EXTI line 18 detect interrupt
CMP1_EXTI_IRQn	CMP1 through EXTI line 19 detect interrupt
CMP2_EXTI_IRQn	CMP2 through EXTI line 20 detect interrupt
CMP3_EXTI_IRQn	CMP3 through EXTI line 21 detect interrupt
SRAMC_ECC_IRQ	SRAMC_ECC interrupt



## nvic\_priority\_group\_set

The description of nvic\_priority\_group\_set is shown as below:

**Table 3-654. 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 3-655. 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 3-653. 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 3-656. 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 3-653. 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 3-657. 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>	NVIC_SystemReset
Input parameter{in}	

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

Example:

```
/* reset the MCU */
nvic_system_reset();
```

### nvic\_vector\_table\_set

The description of nvic\_vector\_table\_set is shown as below:

**Table 3-658. Function nvic\_vector\_table\_set**

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

Example:

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

### system\_lowpower\_set

The description of system\_lowpower\_set is shown as below:

**Table 3-659. Function system\_lowpower\_set**

<b>Function name</b>	system_lowpower_set
<b>Function prototype</b>	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
<b>SCB_LPM_SLEEP_EXIT_ISR</b>	if chose this para, the system always enter low power mode by exiting from ISR
<b>SCB_LPM_DEEPSLEEP_P</b>	if chose this para, the system will enter the DEEPSLEEP mode
<b>SCB_LPM_WAKE_BY_ALL_INT</b>	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 3-660. 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
<b>SCB_LPM_SLEEP_EXIT_ISR</b>	if chose this para, the system will exit low power mode by exiting from ISR
<b>SCB_LPM_DEEPSLEEP_P</b>	if chose this para, the system will enter the SLEEP mode
<b>SCB_LPM_WAKE_BY_ALL_INT</b>	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\_clksource\_set

The description of systick\_clksource\_set is shown as below:

**Table 3-661. Function systick\_clksource\_set**

<b>Function name</b>	systick_clksource_set
<b>Function prototype</b>	void systick_clksource_set(uint32_t systick_clksource);
<b>Function descriptions</b>	set the systick clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>systick_clksource</b>	the systick clock source needed to choose
<i>SYSTICK_CLKSOURC E_HCLK</i>	systick clock source is from HCLK
<i>SYSTICK_CLKSOURC E_HCLK_DIV8</i>	systick clock source is from HCLK/8
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* systick clock source is HCLK/8 */
```

```
systick_clksource_set(SYSTICK_CLKSOURCE_HCLK_DIV8);
```

## 3.21. 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 [3.21.1](#), the PMU firmware functions are introduced in chapter [3.21.2](#).

### 3.21.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

**Table 3-662. PMU Registers**

Registers	Descriptions
PMU_CTL	control register
PMU_CS	control and status register
PMU_LVD1CTL	low voltage detector 1 control register
PMU_LVD2CTL	low voltage detector 2 control register

### 3.21.2. Descriptions of Peripheral functions

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

**Table 3-663. PMU firmware function**

Function name	Function description
pmu_deinit	reset PMU registers
pmu_to_sleepmode	PMU work at sleep mode
pmu_to_deepsleepmode	PMU work at deepsleep mode
pmu_to_standbymode	pmu work at standby mode
pmu_deepsleepmode_vcore_output	PMU work in deepsleep mode ldo voltage output select
pmu_wakeup_pin_enable	enable PMU wakeup pin
pmu_wakeup_pin_disable	disable PMU wakeup pin
pmu_lvd_enable	pmu ldo bypass mode configure
pmu_lvd_disable	ldo output voltage select
pmu_lvd_interrupt_select	enable low voltage detector
pmu_lvd_interrupt_type	disable low voltage detector
pmu_lvd_interrupt_reset_enable	select the interrupt generation condition for the low voltage detector
pmu_lvd_interrupt_reset_disable	select the interrupt type for the low voltage detector
pmu_lvd_output_enable	enable low voltage detector interrupt or reset
pmu_lvd_output_disable	disable low voltage detector interrupt or reset
pmu_lvd_mode_select	enable low voltage detector comparison result output
pmu_lvd_reset_select	disable low voltage detector comparison result output
pmu_lvd_level_select	select low voltage detector mode
pmu_lvd_digital_filter_enable	select low voltage detector reset negation
pmu_lvd_digital_filter_disable	select low voltage detector level
pmu_lvd_sample_clock_select	enable low voltage detector digital filter
pmu_flag_get	disable low voltage detector digital filter
pmu_flag_clear	select low voltage detector sampling clock

#### pmu\_deinit

The description of pmu\_deinit is shown as below:

**Table 3-664. 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 registers */
```

```
pmu_deinit();
```

### pmu\_to\_sleepmode

The description of pmu\_to\_sleepmode is shown as below:

**Table 3-665. Function pmu\_to\_sleepmode**

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

Example:

```
/* PMU work at sleep mode */
```

```
pmu_to_sleepmode(WFI_CMD);
```

### pmu\_to\_deepsleepmode

The description of pmu\_to\_deepsleepmode is shown as below:

**Table 3-666. Function pmu\_to\_deepsleepmode**

Function name	pmu_to_deepsleepmode
Function prototype	void pmu_to_deepsleepmode(uint8_t deepsleepmodecmd)
Function descriptions	PMU work at deepsleep mode
Precondition	-
The called functions	-
Input parameter{in}	
deepsleepmodecmd	command to enter deep sleep mode
WFI_CMD	use WFI command
WFE_CMD	use WFE command

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* PMU work at deepsleep mode */
pmu_to_deepsleepmode(WFI_CMD);
```

### pmu\_to\_standbymode

The description of pmu\_to\_standbymode is shown as below:

**Table 3-667. Function pmu\_to\_standbymode**

Function name	pmu_to_standbymode
Function prototype	void pmu_to_standbymode(void)
Function descriptions	pmu work at standby mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### pmu\_deepsleepmode\_vcore\_output

The description of pmu\_deepsleepmode\_vcore\_output is shown as below:

**Table 3-668. Function pmu\_deepsleepmode\_vcore\_output**

Function name	pmu_deepsleepmode_vcore_output
Function prototype	void pmu_deepsleepmode_vcore_output(uint32_t dsldovs)
Function descriptions	PMU work in deepsleep mode ldo voltage output select
Precondition	-
The called functions	-
Input parameter{in}	
dsldovs	Deep-sleep mode voltage scaling selection
PMU_DSLDOVS_0	LDO voltage is 0.9V
PMU_DSLDOVS_1	LDO voltage is 1.0V



<i>PMU_DSLDOVS_2</i>	LDO voltage is 1.1V
<i>PMU_DSLDOVS_3</i>	LDO voltage is 1.2V
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* Deep-sleep mode voltage scaling selection */
```

```
pmu_deepsleepmode_vcore_output(PMU_DSLDOVS_0);
```

### pmu\_wakeup\_pin\_enable

The description of pmu\_wakeup\_pin\_enable is shown as below:

**Table 3-669. Function pmu\_wakeup\_pin\_enable**

<b>Function name</b>	pmu_wakeup_pin_enable
<b>Function prototype</b>	void pmu_wakeup_pin_enable(void)
<b>Function descriptions</b>	enable PMU wakeup pin
<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 wakeup pin */
```

```
pmu_wakeup_pin_enable();
```

### pmu\_wakeup\_pin\_disable

The description of pmu\_wakeup\_pin\_disable is shown as below:

**Table 3-670. Function pmu\_wakeup\_pin\_disable**

<b>Function name</b>	pmu_wakeup_pin_disable
<b>Function prototype</b>	void pmu_wakeup_pin_disable(void)
<b>Function descriptions</b>	disable PMU wakeup pin
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable PMU wakeup pin */
pmu_wakeup_pin_disable();
```

### pmu\_lvd\_enable

The description of pmu\_lvd\_enable is shown as below:

**Table 3-671. Function pmu\_lvd\_enable**

Function name	pmu_lvd_enable
Function prototype	void pmu_lvd_enable(uint8_t lvd_select)
Function descriptions	enable low voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
lvd_select	select low voltage detector
PMU_LVD_1	select low voltage detector 1
PMU_LVD_2	select low voltage detector 2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable LVD1 */
pmu_lvd_enable(PMU_LVD_1);
```

### pmu\_lvd\_disable

The description of pmu\_lvd\_disable is shown as below:

**Table 3-672. Function pmu\_lvd\_disable**

Function name	pmu_lvd_disable
Function prototype	void pmu_lvd_disable(uint8_t lvd_select)
Function descriptions	disable low voltage detector
Precondition	-
The called functions	-
Input parameter{in}	
lvd_select	select low voltage detector

<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable LVD1 */
pmu_lvd_disable(PMU_LVD_1);
```

### pmu\_lvd\_interrupt\_select

The description of pmu\_lvd\_interrupt\_select is shown as below:

**Table 3-673. Function pmu\_lvd\_interrupt\_select**

<b>Function name</b>	pmu_lvd_interrupt_select
<b>Function prototype</b>	void pmu_lvd_interrupt_select(uint8_t lvd_select, uint32_t select_condition)
<b>Function descriptions</b>	select the interrupt generation condition for the low voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Input parameter{in}</b>	
<b>select_condition</b>	Low voltage detector interrupt trigger mode
<i>PMU_LVD_INT_RISING</i>	vcc >= vdetx(rising)
<i>PMU_LVD_INT_FALLING</i>	vcc < vdetx(falling)
<i>PMU_LVD_INT_BOTH</i>	vcc >= vdetx(rising) or vcc < vdetx(falling)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select the interrupt generation condition for the LVD1 */
pmu_lvd_interrupt_select(PMU_LVD_1, PMU_LVD_INT_RISING);
```

## pmu\_lvd\_interrupt\_type

The description of pmu\_lvd\_interrupt\_type is shown as below:

**Table 3-674. Function pmu\_lvd\_interrupt\_type**

<b>Function name</b>	pmu_lvd_interrupt_type
<b>Function prototype</b>	void pmu_lvd_interrupt_type(uint8_t lvd_select, uint32_t interrupt_type)
<b>Function descriptions</b>	select the interrupt type for the low voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Input parameter{in}</b>	
<b>interrupt_type</b>	Low voltage detector Interrupt Type Select
<i>PMU_LVD_INT_MASKABLE</i>	maskable interrupt
<i>PMU_LVD_INT_NON_MASKABLE</i>	non-maskable interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select the interrupt type for the LVD1 */
pmu_lvd_interrupt_type(PMU_LVD_1, PMU_LVD_INT_MASKABLE);
```

## pmu\_lvd\_interrupt\_reset\_enable

The description of pmu\_lvd\_interrupt\_reset\_enable is shown as below:

**Table 3-675. Function pmu\_lvd\_interrupt\_reset\_enable**

<b>Function name</b>	pmu_lvd_interrupt_reset_enable
<b>Function prototype</b>	void pmu_lvd_interrupt_reset_enable(uint8_t lvd_select)
<b>Function descriptions</b>	enable low voltage detector interrupt or reset
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* enable low voltage detector interrupt or reset */
```

```
pmu_lvd_interrupt_reset_enable(PMU_LVD_1);
```

### pmu\_lvd\_interrupt\_reset\_disable

The description of pmu\_lvd\_interrupt\_reset\_disable is shown as below:

**Table 3-676. Function pmu\_lvd\_interrupt\_reset\_disable**

<b>Function name</b>	pmu_lvd_interrupt_reset_disable
<b>Function prototype</b>	void pmu_lvd_interrupt_reset_disable(uint8_t lvd_select)
<b>Function descriptions</b>	disable low voltage detector interrupt or reset
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable low voltage detector interrupt or reset */
```

```
pmu_lvd_interrupt_reset_disable(PMU_LVD_1);
```

### pmu\_lvd\_output\_enable

The description of pmu\_lvd\_output\_enable is shown as below:

**Table 3-677. Function pmu\_lvd\_output\_enable**

<b>Function name</b>	pmu_lvd_output_enable
<b>Function prototype</b>	void pmu_lvd_output_enable(uint8_t lvd_select)
<b>Function descriptions</b>	enable low voltage detector comparison result output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1

<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable LVD1 comparison result output */
```

```
pmu_lvd_output_enable(PMU_LVD_1);
```

### **pmu\_lvd\_output\_disable**

The description of pmu\_lvd\_output\_disable is shown as below:

**Table 3-678. Function pmu\_lvd\_output\_disable**

<b>Function name</b>	pmu_lvd_output_disable
<b>Function prototype</b>	void pmu_lvd_output_disable(uint8_t lvd_select)
<b>Function descriptions</b>	disable low voltage detector comparison result output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable LVD1 comparison result output */
```

```
pmu_lvd_output_disable(PMU_LVD_1);
```

### **pmu\_lvd\_mode\_select**

The description of pmu\_lvd\_mode\_select is shown as below:

**Table 3-679. Function pmu\_lvd\_mode\_select**

<b>Function name</b>	pmu_lvd_mode_select
<b>Function prototype</b>	void pmu_lvd_mode_select(uint8_t lvd_select, uint32_t mode)
<b>Function descriptions</b>	select low voltage detector mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Input parameter{in}</b>	
<b>mode</b>	low voltage detector mode
<i>PMU_LVD_RESET</i>	low voltage detector reset occurs when the voltage falls below Vdet
<i>PMU_LVD_INTERRUPT</i> <i>T</i>	low voltage detector interrupt occurs when the voltage falls below Vdet
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select low voltage detector mode */
```

```
pmu_lvd_mode_select(PMU_LVD_1, PMU_LVD_RESET);
```

### pmu\_lvd\_reset\_select

The description of pmu\_lvd\_reset\_select is shown as below:

**Table 3-680. Function pmu\_lvd\_reset\_select**

<b>Function name</b>	pmu_lvd_reset_select
<b>Function prototype</b>	void pmu_lvd_reset_select(uint8_t lvd_select, uint32_t mode)
<b>Function descriptions</b>	select low voltage detector reset negation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Input parameter{in}</b>	
<b>mode</b>	low voltage detector reset signal negation mode
<i>PMU_LVD_RESET_AS</i> <i>SERTION</i>	after the LVD reset signal is asserted, the reset signal will be negated after a stabilization time (tLVD)
<i>PMU_LVD_RESET_RIS</i> <i>E</i>	the reset signal will be negated after a stabilization time (tLVDx) following the detection of VCC > Vdet
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select low voltage detector reset negation */
```

```
pmu_lvd_reset_select(PMU_LVD_1, PMU_LVD_RESET_RISE);
```

## pmu\_lvd\_level\_select

The description of pmu\_lvd\_level\_select is shown as below:

**Table 3-681. Function pmu\_lvd\_level\_select**

<b>Function name</b>	pmu_lvd_level_select
<b>Function prototype</b>	void pmu_lvd_level_select(uint8_t lvd_select, uint32_t level)
<b>Function descriptions</b>	select low voltage detector threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Input parameter{in}</b>	
<b>level</b>	low voltage detector threshold
<i>PMU_LVDT_0</i>	low voltage detector level is 4.62v/4.54v
<i>PMU_LVDT_1</i>	low voltage detector level is 4.55v/4.47v
<i>PMU_LVDT_2</i>	low voltage detector level is 4.40v/4.32v
<i>PMU_LVDT_3</i>	low voltage detector level is 3.03v/2.95v
<i>PMU_LVDT_4</i>	low voltage detector level is 2.96v/2.88v
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select low voltage detector threshold */
```

```
pmu_lvd_level_select(PMU_LVD_1, PMU_LVDT_0);
```

## pmu\_lvd\_digital\_filter\_enable

The description of pmu\_lvd\_digital\_filter\_enable is shown as below:

**Table 3-682. Function pmu\_lvd\_digital\_filter\_enable**

<b>Function name</b>	pmu_lvd_digital_filter_enable
<b>Function prototype</b>	void pmu_lvd_digital_filter_enable(uint8_t lvd_select)
<b>Function descriptions</b>	enable low voltage detector digital filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1



<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable LVD1 digital filter */
pmu_lvd_digital_filter_enable(PMU_LVD_1);
```

### **pmu\_lvd\_digital\_filter\_disable**

The description of pmu\_lvd\_digital\_filter\_disable is shown as below:

**Table 3-683. Function pmu\_lvd\_digital\_filter\_disable**

<b>Function name</b>	pmu_lvd_digital_filter_disable
<b>Function prototype</b>	void pmu_lvd_digital_filter_disable(uint8_t lvd_select)
<b>Function descriptions</b>	disable low voltage detector digital filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable LVD1 digital filter */
pmu_lvd_digital_filter_disable(PMU_LVD_1);
```

### **pmu\_lvd\_sample\_clock\_select**

The description of pmu\_lvd\_sample\_clock\_select is shown as below:

**Table 3-684. Function pmu\_lvd\_sample\_clock\_select**

<b>Function name</b>	pmu_lvd_sample_clock_select
<b>Function prototype</b>	void pmu_lvd_sample_clock_select(uint8_t lvd_select, uint32_t clock)
<b>Function descriptions</b>	select low voltage detector sampling clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>lvd_select</b>	select low voltage detector
<i>PMU_LVD_1</i>	select low voltage detector 1
<i>PMU_LVD_2</i>	select low voltage detector 2
<b>Input parameter{in}</b>	
<b>clock</b>	low voltage detector sampling clock
<i>PMU_LVDSAMP_DIV2</i>	low voltage detector sampling clock select 1/2 pclk1 frequency
<i>PMU_LVDSAMP_DIV4</i>	low voltage detector sampling clock select 1/4 pclk1 frequency
<i>PMU_LVDSAMP_DIV8</i>	low voltage detector sampling clock select 1/8 pclk1 frequency
<i>PMU_LVDSAMP_DIV16</i>	low voltage detector sampling clock select 1/16 pclk1 frequency
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select low voltage detector sampling clock */
```

```
pmu_lvd_sample_clock_select(PMU_LVD_1, PMU_LVDSAMP_DIV2);
```

### pmu\_flag\_get

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

**Table 3-685. Function pmu\_flag\_get**

<b>Function name</b>	pmu_flag_get
<b>Function prototype</b>	FlagStatus pmu_flag_get(uint32_t flag)
<b>Function descriptions</b>	get flag state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	flag
<i>PMU_FLAG_WAKEUP</i>	wakeup flag
<i>PMU_FLAG_STANDBY</i>	standby flag
<i>PMU_FLAG_LVD1DET</i>	voltage monitoring 1 voltage change detection flag
<i>PMU_FLAG_LVD1MON</i>	voltage monitoring 1 signal monitor flag
<i>PMU_FLAG_LVD2DET</i>	voltage monitoring 2 voltage change detection flag
<i>PMU_FLAG_LVD2MON</i>	voltage monitoring 2 voltage change detection flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

```
pmu_flag_get(PMU_FLAG_WAKEUP);
```

## pmu\_flag\_clear

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

**Table 3-686. Function pmu\_flag\_clear**

<b>Function name</b>	pmu_flag_clear
<b>Function prototype</b>	void pmu_flag_clear(uint32_t flag)
<b>Function descriptions</b>	clear flag bit
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	xxxxxxxxxxxxxxxxxxxx
PMU_FLAG_RESET_WAKEUP	reset wakeup flag
PMU_FLAG_RESET_STANDBY	reset standby flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear flag state */

pmu_flag_clear(PMU_FLAG_RESET_WAKEUP);
```

## 3.22. POC

The POC (Port Output Controller) can be used to disable pin outputs from the TIMER and GPTIMER channels in a variety of situations. The disabled output pins can choose high-impedance state (Hi-Z) output or general purpose I/O (GPIO) control output. The POC registers are listed in chapter [3.22.1](#), the POC firmware functions are introduced in chapter [3.22.2](#).

### 3.22.1. Descriptions of Peripheral registers

POC registers are listed in the table shown as below:

**Table 3-687. POC registers**

Register	Descriptions
POC_STAT0	POC status register 0
POC_INnDCFG	POC input n detection configuration register
POC_CTL0	POC control register 0

POC_SWDRG	POC software disabling request generation register
POC_CDCFG0	POC complementary detection configuration register 0
POC_CDCFG1	POC complementary detection configuration register 1
POC_ODMODE0	POC output disable mode register 0
POC_ODMODE1	POC output disable mode register 1
POC_REQSEL0	POC request selection register 0
POC_REQSEL1	POC request selection register 1
POC_STAT1	POC status register 1
POC_CTL1	POC control register 1
POC_EXTCTL0	POC extended control register 0
POC_EXTCTL1	POC extended control register 1
POC_INnDMK	POC input n detection mask register
POC_CMPnDMK	POC comparator n detection mask register

### 3.22.2. Descriptions of Peripheral functions

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

**Table 3-688. POC firmware function**

Function name	Function description
poc_deinit	deinitialize POC
poc_input_detection_config	configure POC_INn input detection
poc_input_dreq_status_config	configure POC_INn disabling request status
poc_input_polarity_config	configure POC_INn input polarity
poc_sys_fault_dreq_status_config	configure system fault disabling request status
poc_software_request_generate	generate software disabling request for timer
poc_software_request_stop	stop software disabling request for timer
poc_complementary_detection_struct_para_init	initialize POC complementary detection struct with a default value
poc_timer0_complementary_detection_config	configure TIMER0 complementary channel detection
poc_timer7_complementary_detection_config	configure TIMER7 complementary channel detection
poc_timer0_output_disable_mode_select	select output disable mode for TIMER0
poc_timer7_output_disable_mode_select	select output disable mode for TIMER7
poc_timer1_output_disable_mode_select	select output disable mode for TIMER1
poc_timer2_output_disable_mode_select	select output disable mode for TIMER2
poc_gptimer0_output_disable_mode_select	select output disable mode for GPTIMER0
poc_gptimer1_output_disable_mode_select	select output disable mode for GPTIMER1
poc_request_struct_para_init	initialize POC request struct with a default value
poc_request_select	request select target timer
poc_cmp_dreq_status_config	configure comparator disabling request status
poc_cmp_dreq_status_extended_config	configure comparator disabling request status for target timer

poc_input_detection_mask	mask POC_INn input detection
poc_cmp_output_detection_mask	mask comparator output detection
poc_flag_get	get the POC flag
poc_flag_clear	clear the POC flag
poc_interrupt_enable	enable POC interrupt
poc_interrupt_disable	disable POC interrupt
poc_interrupt_flag_get	get POC interrupt flag
poc_interrupt_flag_clear	clear POC interrupt flag

## Structure poc\_request\_struct

**Table 3-689. Structure poc\_request\_struct**

Member name	Function description
req_pocin0	request from POC_IN0 pin input detection
req_pocin1	request from POC_IN1 pin input detection
req_pocin2	request from POC_IN2 pin input detection
req_pocin3	request from POC_IN3 pin input detection
req_pocin4	request from POC_IN4 pin input detection
req_pocin5	request from POC_IN5 pin input detection
req_comparator	request from comparator output detection

## Structure poc\_complementary\_detection\_struct

**Table 3-690. Structure poc\_complementary\_detection\_struct**

Member name	Function description
ccdreqstatus	concurrent conduction disabling request status
polarityselen	timer polarity selection enable
mch2polarity	multi mode channel 2 active polarity selection
ch2polarity	channel 2 active polarity selection
mch1polarity	multi mode channel 1 active polarity selection
ch1polarity	channel 1 active polarity selection
mch0polarity	multi mode channel 0 active polarity selection
ch0polarity	channel 0 active polarity selection

## Enum poc\_pin\_enum

**Table 3-691. Enum poc\_pin\_enum**

Member name	Function description
POC_IN0	POC_IN0 input pin
POC_IN1	POC_IN1 input pin
POC_IN2	POC_IN2 input pin
POC_IN3	POC_IN3 input pin
POC_IN4	POC_IN4 input pin
POC_IN5	POC_IN5 input pin

## Enum cmp\_output\_enum

Table 3-692. Enum cmp\_output\_enum

Member name	Function description
CMP0_OUT	comparator 0 output
CMP1_OUT	comparator 1 output
CMP2_OUT	comparator 2 output
CMP3_OUT	comparator 3 output

## Enum target\_timer\_enum

Table 3-693. Enum target\_timer\_enum

Member name	Function description
TARGET_TIMER0	TIMER0 is target timer
TARGET_TIMER7	TIMER7 is target timer
TARGET_TIMER1	TIMER1 is target timer
TARGET_TIMER2	TIMER2 is target timer
TARGET_GPTIMER0	GPTIMER0 is target timer
TARGET_GPTIMER1	GPTIMER1 is target timer

## Enum poc\_interrupt\_enum

Table 3-694. Enum poc\_interrupt\_enum

Member name	Function description
POC_INT_IN0	POC_IN0 input detection interrupt
POC_INT_IN1	POC_IN1 input detection interrupt
POC_INT_IN2	POC_IN2 input detection interrupt
POC_INT_IN3	POC_IN3 input detection interrupt
POC_INT_IN4	POC_IN4 input detection interrupt
POC_INT_IN5	POC_IN5 input detection interrupt
POC_INT_TIMER0CC	TIMER0 concurrent conduction detection interrupt
POC_INT_TIMER7CC	TIMER7 concurrent conduction detection interrupt

## poc\_deinit

The description of poc\_deinit is shown as below:

Table 3-695. Function poc\_deinit

Function name	poc_deinit
Function prototype	void poc_deinit(void)
Function descriptions	deinitialize POC
Precondition	-
The called functions	rcu_periph_reset_enable/ rcu_periph_reset_disable
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* deinitialize POC */
```

```
dac_deinit();
```

### poc\_input\_detection\_config

The description of poc\_input\_detection\_config is shown as below:

**Table 3-696. Function poc\_input\_detection\_config**

Function name	poc_input_detection_config
Function prototype	void poc_input_detection_config(poc_pin_enum poc_pin, uint32_t detection_mode, uint32_t sampling_number)
Function descriptions	configure POC_INn input detection
Precondition	-
The called functions	-
Input parameter{in}	
poc_pin	refer to enum <a href="#">Table 3-691. Enum poc_pin_enum</a>
Input parameter{in}	
detection_mode	input detection mode
POC_EDGE_DETECTI N_DIV1	sampling frequency is $f_{HCLK}$ and generate a request on the falling edge (INPL=0) or rising edge (INPL=1)
POC_LEVEL_DETECTI ON_DIV8	sampling frequency is $f_{HCLK}/8$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
POC_LEVEL_DETECTI ON_DIV16	sampling frequency is $f_{HCLK}/16$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
POC_LEVEL_DETECTI ON_DIV128	sampling frequency is $f_{HCLK}/128$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
POC_LEVEL_DETECTI ON_DIV1	sampling frequency is $f_{HCLK}$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
POC_LEVEL_DETECTI ON_DIV2	sampling frequency is $f_{HCLK}/2$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
POC_LEVEL_DETECTI ON_DIV4	sampling frequency is $f_{HCLK}/4$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
POC_LEVEL_DETECTI ON_DIV256	sampling frequency is $f_{HCLK}/256$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain

	number of times
<i>POC_LEVEL_DETECTI ON_DIV512</i>	sampling frequency is $f_{HCLK}/512$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
<i>POC_LEVEL_DETECTI ON_DIV32</i>	sampling frequency is $f_{HCLK}/32$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
<i>POC_LEVEL_DETECTI ON_DIV64</i>	sampling frequency is $f_{HCLK}/64$ and generate a request when the low level (INPL=0) or high level (INPL=1) is continuously sampled a certain number of times
<i>POC_EDGE_DETECTIO N_DIV2</i>	sampling frequency is $f_{HCLK}/2$ and generate a request on the falling edge (INPL=0) or rising edge (INPL=1)
<i>POC_EDGE_DETECTIO N_DIV4</i>	sampling frequency is $f_{HCLK}/4$ and generate a request on the falling edge (INPL=0) or rising edge (INPL=1)
<b>Input parameter{in}</b>	
<b>sampling_number</b>	valid sampling number
<i>POC_SAMPLING_NUM_ 16_TIMES</i>	sampling number is 16 times
<i>POC_SAMPLING_NUM_ 4_TIMES</i>	sampling number is 4 times
<i>POC_SAMPLING_NUM_ 8_TIMES</i>	sampling number is 8 times
<i>POC_SAMPLING_NUM_ 9_TIMES</i>	sampling number is 9 times
<i>POC_SAMPLING_NUM_ 10_TIMES</i>	sampling number is 10 times
<i>POC_SAMPLING_NUM_ 11_TIMES</i>	sampling number is 11 times
<i>POC_SAMPLING_NUM_ 12_TIMES</i>	sampling number is 12 times
<i>POC_SAMPLING_NUM_ 13_TIMES</i>	sampling number is 13 times
<i>POC_SAMPLING_NUM_ 14_TIMES</i>	sampling number is 14 times
<i>POC_SAMPLING_NUM_ 15_TIMES</i>	sampling number is 15 times
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure POC_IN0 input detection */
```



```
poc_input_detection_config(POC_IN0, POC_LEVEL_DETECTION_DIV8,
POC_SAMPLING_NUM_16_TIMES);
```

### poc\_input\_dreq\_status\_config

The description of poc\_input\_dreq\_status\_config is shown as below:

**Table 3-697. Function poc\_input\_dreq\_status\_config**

<b>Function name</b>	poc_input_dreq_status_config
<b>Function prototype</b>	void poc_input_dreq_status_config(poc_pin_enum poc_pin, uint32_t indreq_status)
<b>Function descriptions</b>	configure POC_INn disabling request status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>poc_pin</b>	refer to enum <a href="#">Table 3-691. Enum poc_pin_enum</a>
<b>Input parameter{in}</b>	
<b>indreq_status</b>	POC_INn disabling request status
<i>POC_INn_DREQ_DISABLE</i>	disabling request generated by POC_INn detection is disabled
<i>POC_INn_DREQ_ENABLE</i>	disabling request generated by POC_INn detection is enabled
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure POC_INn disabling request status */
poc_input_dreq_status_config(POC_IN0, POC_INn_DREQ_ENABLE);
```

### poc\_input\_polarity\_config

The description of poc\_input\_polarity\_config is shown as below:

**Table 3-698. Function poc\_input\_polarity\_config**

<b>Function name</b>	poc_input_polarity_config
<b>Function prototype</b>	void poc_input_polarity_config(poc_pin_enum poc_pin, uint32_t input_polarity)
<b>Function descriptions</b>	configure POC_INn input polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>poc_pin</b>	refer to enum <a href="#">Table 3-691. Enum poc_pin_enum</a>
<b>Input parameter{in}</b>	

<b>input_polarity</b>	POC_INn input polarity
<i>POC_INPUT_POLARITY_NONINVERTED</i>	POC_INn input not inverted
<i>POC_INPUT_POLARITY_INVERTED</i>	POC_INn input inverted
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure POC_IN0 input polarity */
```

```
poc_input_polarity_config(POC_IN0, POC_INPUT_POLARITY_INVERTED);
```

### **poc\_sys\_fault\_dreq\_status\_config**

The description of poc\_sys\_fault\_dreq\_status\_config is shown as below:

**Table 3-699. Function poc\_sys\_fault\_dreq\_status\_config**

<b>Function name</b>	poc_sys_fault_dreq_status_config
<b>Function prototype</b>	void poc_sys_fault_dreq_status_config(uint32_t hxtalsdreq_status, uint32_t lockupdreq_status)
<b>Function descriptions</b>	configure system fault disabling request status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hxtalsdreq_status</b>	HXTAL stuck disabling request status
<i>HXTALS_DREQ_DISABLE</i>	HXTAL stuck disabling request disable
<i>HXTALS_DREQ_ENABLE</i>	HXTAL stuck disabling request enable
<b>Input parameter{in}</b>	
<b>lockupdreq_status</b>	CPU LOCKUP disabling request status
<i>LOCKUP_DREQ_DISABLE</i>	CPU LOCKUP disabling request disable
<i>LOCKUP_DREQ_ENABLE</i>	CPU LOCKUP disabling request enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure system fault disabling request status */
```

```
poc_sys_fault_dreq_status_config(HXTALS_DREQ_ENABLE, LOCKUP_DREQ_ENABLE);
```

### poc\_software\_request\_generate

The description of poc\_software\_request\_generate is shown as below:

**Table 3-700. Function poc\_software\_request\_generate**

<b>Function name</b>	poc_software_request_generate
<b>Function prototype</b>	void poc_software_request_generate(target_timer_enum target_timer)
<b>Function descriptions</b>	generate software disabling request for timer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_timer</b>	refer to enum <a href="#">Table 3-693. Enum target timer enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* generate software disabling request for TIMER0 */
poc_software_request_generate(TARGET_TIMER0);
```

### poc\_software\_request\_stop

The description of poc\_software\_request\_stop is shown as below:

**Table 3-701. Function poc\_software\_request\_stop**

<b>Function name</b>	poc_software_request_stop
<b>Function prototype</b>	void poc_software_request_stop(target_timer_enum target_timer)
<b>Function descriptions</b>	stop software disabling request for timer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_timer</b>	refer to enum <a href="#">Table 3-693. Enum target timer enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* stop software disabling request for TIMER0 */
poc_software_request_stop(TARGET_TIMER0);
```

## poc\_complementary\_detection\_struct\_para\_init

The description of poc\_complementary\_detection\_struct\_para\_init is shown as below:

**Table 3-702. Function poc\_complementary\_detection\_struct\_para\_init**

Function name	poc_complementary_detection_struct_para_init
Function prototype	void poc_complementary_detection_struct_para_init(poc_complementary_detection_struct *detpara)
Function descriptions	initialize POC complementary detection struct with a default value
Precondition	-
The called functions	-
Input parameter{in}	
detpara	POC complementary detection struct parameter, the structure members can refer to <a href="#">Table 3-702. Function poc_complementary_detection_struct_para_init</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize POC complementary detection struct with a default value */
```

```
poc_complementary_detection_struct para;
```

```
poc_complementary_detection_struct_para_init(&para);
```

## poc\_timer0\_complementary\_detection\_config

The description of poc\_timer0\_complementary\_detection\_config is shown as below:

**Table 3-703. Function poc\_timer0\_complementary\_detection\_config**

Function name	poc_timer0_complementary_detection_config
Function prototype	void poc_timer0_complementary_detection_config(poc_complementary_detection_struct *detpara)
Function descriptions	configure TIMER0 complementary channel detection
Precondition	-
The called functions	-
Input parameter{in}	
detpara	POC complementary detection struct parameter, the structure members can refer to <a href="#">Table 3-702. Function poc_complementary_detection_struct_para_init</a> .
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* configure TIMER0 complementary channel detection */
```

```
poc_complementary_detection_struct para;
```

```
poc_timer0_complementary_detection_config(&para);
```

### **poc\_timer7\_complementary\_detection\_config**

The description of poc\_timer7\_complementary\_detection\_config is shown as below:

**Table 3-704. Function poc\_timer7\_complementary\_detection\_config**

Function name	poc_timer7_complementary_detection_config
Function prototype	void poc_timer7_complementary_detection_config(poc_complementary_detection_struct *detpara)
Function descriptions	configure TIMER7 complementary channel detection
Precondition	-
The called functions	-
Input parameter{in}	
detpara	POC complementary detection struct parameter, the structure members can refer to <a href="#">Table 3-702. Function poc_complementary_detection_struct para init.</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER7 complementary channel detection */
```

```
poc_complementary_detection_struct para;
```

```
poc_timer7_complementary_detection_config(&para);
```

### **poc\_timer0\_output\_disable\_mode\_select**

The description of poc\_timer0\_output\_disable\_mode\_select is shown as below:

**Table 3-705. Function poc\_timer0\_output\_disable\_mode\_select**

Function name	poc_timer0_output_disable_mode_select
Function prototype	void poc_timer0_output_disable_mode_select(uint8_t ch0mch0_pin_mode, uint8_t ch1mch1_pin_mode, uint8_t ch2mch2_pin_mode, uint8_t ch3mch3_pin_mode)

<b>Function descriptions</b>	select output disable mode for TIMER0
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ch0mch0_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch1mch1_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch2mch2_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch3mch3_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select output disable mode for TIMER0 */
```

```
poc_timer0_output_disable_mode_select(POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT,          POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT);
```

## poc\_timer7\_output\_disable\_mode\_select

The description of poc\_timer7\_output\_disable\_mode\_select is shown as below:

**Table 3-706. Function poc\_timer7\_output\_disable\_mode\_select**

<b>Function name</b>	poc_timer7_output_disable_mode_select
<b>Function prototype</b>	void poc_timer7_output_disable_mode_select(uint8_t ch0mch0_pin_mode, uint8_t ch1mch1_pin_mode, uint8_t ch2mch2_pin_mode, uint8_t ch3mch3_pin_mode)
<b>Function descriptions</b>	select output disable mode for TIMER7
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ch0mch0_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch1mch1_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch2mch2_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch3mch3_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select output disable mode for TIMER7 */
```

```
poc_timer7_output_disable_mode_select(POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT,                POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT);
```

### **poc\_timer1\_output\_disable\_mode\_select**

The description of poc\_timer1\_output\_disable\_mode\_select is shown as below:

**Table 3-707. Function poc\_timer1\_output\_disable\_mode\_select**

Function name	poc_timer1_output_disable_mode_select
Function prototype	void poc_timer1_output_disable_mode_select(uint8_t ch0_pin_mode, uint8_t ch1_pin_mode, uint8_t ch2_pin_mode, uint8_t ch3_pin_mode)
Function descriptions	select output disable mode for TIMER1
Precondition	-
The called functions	-
Input parameter{in}	
ch0_pin_mode	pin output mode
POC_TIMER_CHANNEL_OUT	timer channel output
POC_HIGH_IMPEDANCE_OUT	High-impedance output
POC_GPIO_OUT	GPIO output
Input parameter{in}	
ch1_pin_mode	pin output mode
POC_TIMER_CHANNEL_OUT	timer channel output
POC_HIGH_IMPEDANCE_OUT	High-impedance output
POC_GPIO_OUT	GPIO output
Input parameter{in}	
ch2_pin_mode	pin output mode
POC_TIMER_CHANNEL_OUT	timer channel output
POC_HIGH_IMPEDANCE_OUT	High-impedance output
POC_GPIO_OUT	GPIO output
Input parameter{in}	
ch3_pin_mode	pin output mode
POC_TIMER_CHANNEL_OUT	timer channel output
POC_HIGH_IMPEDANCE_OUT	High-impedance output



<i>E_OUT</i>	
<i>POC_GPIO_OUT</i>	GPIO output
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select output disable mode for TIMER1 */
```

```
poc_timer1_output_disable_mode_select(POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT,                POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT);
```

### **poc\_timer2\_output\_disable\_mode\_select**

The description of poc\_timer2\_output\_disable\_mode\_select is shown as below:

**Table 3-708. Function poc\_timer2\_output\_disable\_mode\_select**

<b>Function name</b>	poc_timer2_output_disable_mode_select
<b>Function prototype</b>	void poc_timer2_output_disable_mode_select(uint8_t ch0_pin_mode, uint8_t ch1_pin_mode, uint8_t ch2_pin_mode, uint8_t ch3_pin_mode)
<b>Function descriptions</b>	select output disable mode for TIMER2
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ch0_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch1_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch2_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output

<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch3_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select output disable mode for TIMER2 */
```

```
poc_timer2_output_disable_mode_select(POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT,          POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT);
```

### **poc\_gptimer0\_output\_disable\_mode\_select**

The description of `poc_gptimer0_output_disable_mode_select` is shown as below:

**Table 3-709. Function `poc_gptimer0_output_disable_mode_select`**

<b>Function name</b>	<code>poc_gptimer0_output_disable_mode_select</code>
<b>Function prototype</b>	<code>void poc_gptimer0_output_disable_mode_select(uint8_t ch0_pin_mode, uint8_t ch1_pin_mode)</code>
<b>Function descriptions</b>	select output disable mode for GPTIMER0
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ch0_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output
<b>Input parameter{in}</b>	
<b>ch1_pin_mode</b>	pin output mode
<i>POC_TIMER_CHANNEL_OUT</i>	timer channel output
<i>POC_HIGH_IMPEDANCE_OUT</i>	High-impedance output
<i>POC_GPIO_OUT</i>	GPIO output

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select output disable mode for GPTIMER0 */
```

```
poc_gptimer0_output_disable_mode_select(POC_HIGH_IMPEDANCE_OUT,
POC_HIGH_IMPEDANCE_OUT);
```

### **poc\_gptimer1\_output\_disable\_mode\_select**

The description of poc\_gptimer1\_output\_disable\_mode\_select is shown as below:

**Table 3-710. Function poc\_gptimer1\_output\_disable\_mode\_select**

Function name	poc_gptimer1_output_disable_mode_select
Function prototype	void poc_gptimer1_output_disable_mode_select(uint8_t ch0_pin_mode, uint8_t ch1_pin_mode)
Function descriptions	select output disable mode for GPTIMER1
Precondition	-
The called functions	-
Input parameter{in}	
ch0_pin_mode	pin output mode
POC_TIMER_CHANNEL_OUT	timer channel output
POC_HIGH_IMPEDANCE_OUT	High-impedance output
POC_GPIO_OUT	GPIO output
Input parameter{in}	
ch1_pin_mode	pin output mode
POC_TIMER_CHANNEL_OUT	timer channel output
POC_HIGH_IMPEDANCE_OUT	High-impedance output
POC_GPIO_OUT	GPIO output
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select output disable mode for GPTIMER1 */
```

```
poc_gptimer1_output_disable_mode_select(POC_HIGH_IMPEDANCE_OUT,
```

POC\_HIGH\_IMPEDANCE\_OUT);

### poc\_request\_struct\_para\_init

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

**Table 3-711. Function poc\_request\_struct\_para\_init**

<b>Function name</b>	poc_request_struct_para_init
<b>Function prototype</b>	void poc_request_struct_para_init(poc_request_struct *request)
<b>Function descriptions</b>	initialize POC request struct with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>request</b>	POC request struct parameter, the structure members can refer to <a href="#">Table 3-689. Structure poc_request_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize POC request struct with a default value */
```

```
poc_request_struct poc_request;
```

```
poc_request_struct_para_init(&poc_request);
```

### poc\_request\_select

The description of poc\_request\_select is shown as below:

**Table 3-712. Function poc\_request\_select**

<b>Function name</b>	poc_request_select
<b>Function prototype</b>	void poc_request_select(poc_request_struct *request, target_timer_enum target_timer)
<b>Function descriptions</b>	request select target timer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>request</b>	POC request struct parameter, the structure members can refer to <a href="#">Table 3-689. Structure poc_request_struct</a> .
<b>Input parameter{in}</b>	
<b>target_timer</b>	refer to enum <a href="#">Table 3-693. Enum target_timer_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* request select TIMER0 */

poc_request_struct poc_request;

poc_request_select(&poc_request, TARGET_TIMER0);
```

### poc\_cmp\_dreq\_status\_config

The description of poc\_cmp\_dreq\_status\_config is shown as below:

**Table 3-713. Function poc\_cmp\_dreq\_status\_config**

Function name	poc_cmp_dreq_status_config
Function prototype	void poc_cmp_dreq_status_config(uint32_t cmp0dreq_status, uint32_t cmp1dreq_status, uint32_t cmp2dreq_status, uint32_t cmp3dreq_status)
Function descriptions	configure comparator disabling request status
Precondition	-
The called functions	-
Input parameter{in}	
cmp0dreq_status	comparator 0 disabling request status
CMP0_DREQ_DISABLE	comparator 0 disabling request disable
CMP0_DREQ_ENABLE	comparator 0 disabling request enable
Input parameter{in}	
cmp1dreq_status	comparator 1 disabling request status
CMP1_DREQ_DISABLE	comparator 1 disabling request disable
CMP1_DREQ_ENABLE	comparator 1 disabling request enable
Input parameter{in}	
cmp2dreq_status	comparator 2 disabling request status
CMP2_DREQ_DISABLE	comparator 2 disabling request disable
CMP2_DREQ_ENABLE	comparator 2 disabling request enable
Input parameter{in}	
cmp1dreq_status	comparator 3 disabling request status
CMP3_DREQ_DISABLE	comparator 3 disabling request disable
CMP3_DREQ_ENABLE	comparator 3 disabling request enable
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure comparator disabling request status */

poc_cmp_dreq_status_config(CMP0_DREQ_DISABLE,          CMP1_DREQ_DISABLE,
CMP2_DREQ_ENABLE, CMP3_DREQ_ENABLE);
```

## poc\_cmp\_dreq\_status\_extended\_config

The description of poc\_cmp\_dreq\_status\_extended\_config is shown as below:

**Table 3-714. Function poc\_cmp\_dreq\_status\_extended\_config**

<b>Function name</b>	poc_cmp_dreq_status_extended_config
<b>Function prototype</b>	void poc_cmp_dreq_status_extended_config(uint32_t cmp0dreq_status, uint32_t cmp1dreq_status, uint32_t cmp2dreq_status, uint32_t cmp3dreq_status, target_timer_enum target_timer)
<b>Function descriptions</b>	configure comparator disabling request status for target timer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp0dreq_status</b>	comparator 0 disabling request status
<i>CMP0_DREQ_DISABLE</i>	comparator 0 disabling request disable
<i>CMP0_DREQ_ENABLE</i>	comparator 0 disabling request enable
<b>Input parameter{in}</b>	
<b>cmp1dreq_status</b>	comparator 1 disabling request status
<i>CMP1_DREQ_DISABLE</i>	comparator 1 disabling request disable
<i>CMP1_DREQ_ENABLE</i>	comparator 1 disabling request enable
<b>Input parameter{in}</b>	
<b>cmp2dreq_status</b>	comparator 2 disabling request status
<i>CMP2_DREQ_DISABLE</i>	comparator 2 disabling request disable
<i>CMP2_DREQ_ENABLE</i>	comparator 2 disabling request enable
<b>Input parameter{in}</b>	
<b>cmp3dreq_status</b>	comparator 3 disabling request status
<i>CMP3_DREQ_DISABLE</i>	comparator 3 disabling request disable
<i>CMP3_DREQ_ENABLE</i>	comparator 3 disabling request enable
<b>Input parameter{in}</b>	
<b>target_timer</b>	refer to enum <a href="#">Table 3-693. Enum target_timer_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure comparator disabling request status for TIMER0 */
```

```
poc_cmp_dreq_status_extended_config(CMP0_DREQ_DISABLE, CMP1_DREQ_DISABLE,
CMP2_DREQ_ENABLE, CMP3_DREQ_ENABLE, TARGET_TIMER0);
```

## poc\_input\_detection\_mask

The description of poc\_input\_detection\_mask is shown as below:

Table 3-715. Function poc\_input\_detection\_mask

Function name	poc_input_detection_mask
Function prototype	void poc_input_detection_mask(poc_pin_enum poc_pin, uint32_t mask_source)
Function descriptions	mask POC_INn input detection
Precondition	-
The called functions	-
Input parameter{in}	
poc_pin	refer to enum <a href="#">Table 3-691. Enum poc_pin_enum</a>
Input parameter{in}	
mask_source	mask source selection
POC_REQUEST_NOT_MASKED	detection is not masked
POC_MASK_SOURCE_TIMER0_CH0	detection is masked by TIMER0_CH0
POC_MASK_SOURCE_TIMER0_CH1	detection is masked by TIMER0_CH1
POC_MASK_SOURCE_TIMER0_CH2	detection is masked by TIMER0_CH2
POC_MASK_SOURCE_TIMER0_CH3	detection is masked by TIMER0_CH3
POC_MASK_SOURCE_TIMER0_MCH0	detection is masked by TIMER0_MCH0
POC_MASK_SOURCE_TIMER0_MCH1	detection is masked by TIMER0_MCH1
POC_MASK_SOURCE_TIMER0_MCH2	detection is masked by TIMER0_MCH2
POC_MASK_SOURCE_TIMER0_MCH3	detection is masked by TIMER0_MCH3
POC_MASK_SOURCE_TIMER7_CH0	detection is masked by TIMER7_CH0
POC_MASK_SOURCE_TIMER7_CH1	detection is masked by TIMER7_CH1
POC_MASK_SOURCE_TIMER7_CH2	detection is masked by TIMER7_CH2
POC_MASK_SOURCE_TIMER7_CH3	detection is masked by TIMER7_CH3
POC_MASK_SOURCE_TIMER7_MCH0	detection is masked by TIMER7_MCH0
POC_MASK_SOURCE_TIMER7_MCH1	detection is masked by TIMER7_MCH1
POC_MASK_SOURCE_TIMER7_MCH2	detection is masked by TIMER7_MCH2

<i>POC_MASK_SOURCE_TIMER7_MCH3</i>	detection is masked by TIMER7_MCH3
<i>POC_MASK_SOURCE_TIMER1_CH0</i>	detection is masked by TIMER1_CH0
<i>POC_MASK_SOURCE_TIMER1_CH1</i>	detection is masked by TIMER1_CH1
<i>POC_MASK_SOURCE_TIMER1_CH2</i>	detection is masked by TIMER1_CH2
<i>POC_MASK_SOURCE_TIMER1_CH3</i>	detection is masked by TIMER1_CH3
<i>POC_MASK_SOURCE_TIMER2_CH0</i>	detection is masked by TIMER2_CH0
<i>POC_MASK_SOURCE_TIMER2_CH1</i>	detection is masked by TIMER2_CH1
<i>POC_MASK_SOURCE_TIMER2_CH2</i>	detection is masked by TIMER2_CH2
<i>POC_MASK_SOURCE_TIMER2_CH3</i>	detection is masked by TIMER2_CH3
<i>POC_MASK_SOURCE_GPTIMER0_CH0</i>	detection is masked by GPTIMER0_CH0
<i>POC_MASK_SOURCE_GPTIMER0_CH1</i>	detection is masked by GPTIMER0_CH1
<i>POC_MASK_SOURCE_GPTIMER1_CH0</i>	detection is masked by GPTIMER1_CH0
<i>POC_MASK_SOURCE_GPTIMER1_CH1</i>	detection is masked by GPTIMER1_CH1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* mask POC_IN0 input detection */
```

```
poc_input_detection_mask(POC_IN0, POC_MASK_SOURCE_TIMER0_CH0);
```

### **poc\_cmp\_output\_detection\_mask**

The description of poc\_cmp\_output\_detection\_mask is shown as below:

**Table 3-716. Function poc\_cmp\_output\_detection\_mask**

<b>Function name</b>	poc_cmp_output_detection_mask
<b>Function prototype</b>	void poc_cmp_output_detection_mask(cmp_output_enum cmp_output, uint32_t mask_source)
<b>Function descriptions</b>	mask comparator output detection



<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_output</b>	refer to enum <a href="#">Table 3-692. Enum cmp_output_enum</a>
<b>Input parameter{in}</b>	
<b>mask_source</b>	mask source selection
<i>POC_REQUEST_NOT_MASKED</i>	detection is not masked
<i>POC_MASK_SOURCE_TIMER0_CH0</i>	detection is masked by TIMER0_CH0
<i>POC_MASK_SOURCE_TIMER0_CH1</i>	detection is masked by TIMER0_CH1
<i>POC_MASK_SOURCE_TIMER0_CH2</i>	detection is masked by TIMER0_CH2
<i>POC_MASK_SOURCE_TIMER0_CH3</i>	detection is masked by TIMER0_CH3
<i>POC_MASK_SOURCE_TIMER0_MCH0</i>	detection is masked by TIMER0_MCH0
<i>POC_MASK_SOURCE_TIMER0_MCH1</i>	detection is masked by TIMER0_MCH1
<i>POC_MASK_SOURCE_TIMER0_MCH2</i>	detection is masked by TIMER0_MCH2
<i>POC_MASK_SOURCE_TIMER0_MCH3</i>	detection is masked by TIMER0_MCH3
<i>POC_MASK_SOURCE_TIMER7_CH0</i>	detection is masked by TIMER7_CH0
<i>POC_MASK_SOURCE_TIMER7_CH1</i>	detection is masked by TIMER7_CH1
<i>POC_MASK_SOURCE_TIMER7_CH2</i>	detection is masked by TIMER7_CH2
<i>POC_MASK_SOURCE_TIMER7_CH3</i>	detection is masked by TIMER7_CH3
<i>POC_MASK_SOURCE_TIMER7_MCH0</i>	detection is masked by TIMER7_MCH0
<i>POC_MASK_SOURCE_TIMER7_MCH1</i>	detection is masked by TIMER7_MCH1
<i>POC_MASK_SOURCE_TIMER7_MCH2</i>	detection is masked by TIMER7_MCH2
<i>POC_MASK_SOURCE_TIMER7_MCH3</i>	detection is masked by TIMER7_MCH3
<i>POC_MASK_SOURCE_TIMER1_CH0</i>	detection is masked by TIMER1_CH0
<i>POC_MASK_SOURCE_TIMER1_CH1</i>	detection is masked by TIMER1_CH1

<i>TIMER1_CH1</i>	
<i>POC_MASK_SOURCE_TIMER1_CH2</i>	detection is masked by <i>TIMER1_CH2</i>
<i>POC_MASK_SOURCE_TIMER1_CH3</i>	detection is masked by <i>TIMER1_CH3</i>
<i>POC_MASK_SOURCE_TIMER2_CH0</i>	detection is masked by <i>TIMER2_CH0</i>
<i>POC_MASK_SOURCE_TIMER2_CH1</i>	detection is masked by <i>TIMER2_CH1</i>
<i>POC_MASK_SOURCE_TIMER2_CH2</i>	detection is masked by <i>TIMER2_CH2</i>
<i>POC_MASK_SOURCE_TIMER2_CH3</i>	detection is masked by <i>TIMER2_CH3</i>
<i>POC_MASK_SOURCE_GPTIMER0_CH0</i>	detection is masked by <i>GPTIMER0_CH0</i>
<i>POC_MASK_SOURCE_GPTIMER0_CH1</i>	detection is masked by <i>GPTIMER0_CH1</i>
<i>POC_MASK_SOURCE_GPTIMER1_CH0</i>	detection is masked by <i>GPTIMER1_CH0</i>
<i>POC_MASK_SOURCE_GPTIMER1_CH1</i>	detection is masked by <i>GPTIMER1_CH1</i>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* mask CMP0 output detection */
```

```
poc_cmp_output_detection_mask(CMP0_OUT, POC_MASK_SOURCE_TIMER0_CH0);
```

### **poc\_flag\_get**

The description of `poc_flag_get` is shown as below:

**Table 3-717. Function `poc_flag_get`**

<b>Function name</b>	<code>poc_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus poc_flag_get(uint32_t flag)</code>
<b>Function descriptions</b>	get the POC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	the POC flag
<i>POC_FLAG_IN0IF</i>	<i>POC_IN0</i> input detection flag
<i>POC_FLAG_IN1IF</i>	<i>POC_IN1</i> input detection flag

<i>POC_FLAG_IN2IF</i>	POC_IN2 input detection flag
<i>POC_FLAG_IN3IF</i>	POC_IN3 input detection flag
<i>POC_FLAG_IN4IF</i>	POC_IN4 input detection flag
<i>POC_FLAG_IN5IF</i>	POC_IN5 input detection flag
<i>POC_FLAG_HXTALSDF</i>	HXTAL stuck detection flag
<i>POC_FLAG_LOCKUPDF</i>	CPU LOCKUP detection flag
<i>POC_FLAG_TIMER0_C CIF</i>	TIMER0 concurrent conduction detection flag
<i>POC_FLAG_TIMER7_C CIF</i>	TIMER7 concurrent conduction detection flag
<i>POC_FLAG_CMP0DF</i>	comparator 0 detection flag
<i>POC_FLAG_CMP1DF</i>	comparator 1 detection flag
<i>POC_FLAG_CMP2DF</i>	comparator 2 detection flag
<i>POC_FLAG_CMP3DF</i>	comparator 3 detection flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
FlagStatus status;
```

```
/* get POC_IN0 input detection flag */
```

```
status = poc_flag_get(POC_FLAG_IN0IF);
```

### **poc\_flag\_clear**

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

**Table 3-718. Function poc\_flag\_clear**

<b>Function name</b>	poc_flag_clear
<b>Function prototype</b>	void poc_flag_clear(uint32_t flag)
<b>Function descriptions</b>	clear the POC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	the POC flag
<i>POC_FLAG_IN0IF</i>	POC_IN0 input detection flag
<i>POC_FLAG_IN1IF</i>	POC_IN1 input detection flag
<i>POC_FLAG_IN2IF</i>	POC_IN2 input detection flag
<i>POC_FLAG_IN3IF</i>	POC_IN3 input detection flag
<i>POC_FLAG_IN4IF</i>	POC_IN4 input detection flag
<i>POC_FLAG_IN5IF</i>	POC_IN5 input detection flag
<i>POC_FLAG_HXTALSDF</i>	HXTAL stuck detection flag

<i>POC_FLAG_LOCKUPDF</i>	CPU LOCKUP detection flag
<i>POC_FLAG_TIMER0_C CIF</i>	TIMER0 concurrent conduction detection flag
<i>POC_FLAG_TIMER7_C CIF</i>	TIMER7 concurrent conduction detection flag
<i>POC_FLAG_CMP0DF</i>	comparator 0 detection flag
<i>POC_FLAG_CMP1DF</i>	comparator 1 detection flag
<i>POC_FLAG_CMP2DF</i>	comparator 2 detection flag
<i>POC_FLAG_CMP3DF</i>	comparator 3 detection flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear POC_IN0 input detection flag */
poc_flag_clear(POC_FLAG_IN0IF);
```

### **poc\_interrupt\_enable**

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

**Table 3-719. Function poc\_interrupt\_enable**

<b>Function name</b>	poc_interrupt_enable
<b>Function prototype</b>	void poc_interrupt_enable(poc_interrupt_enum interrupt)
<b>Function descriptions</b>	enable POC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	refer to enum <a href="#">Table 3-694. Enum poc_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable POC_IN0 input detection interrupt */
poc_interrupt_enable(POC_INT_IN0);
```

### **poc\_interrupt\_disable**

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

Table 3-720. Function poc\_interrupt\_disable

Function name	poc_interrupt_disable
Function prototype	void poc_interrupt_disable(poc_interrupt_enum interrupt)
Function descriptions	disable POC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	refer to enum <a href="#">Table 3-694. Enum poc_interrupt_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable POC_IN0 input detection interrupt */
poc_interrupt_disable(POC_INT_IN0);
```

### poc\_interrupt\_flag\_get

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

Table 3-721. Function poc\_interrupt\_flag\_get

Function name	poc_interrupt_flag_get
Function prototype	FlagStatus poc_interrupt_flag_get(uint32_t int_flag)
Function descriptions	get POC interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	the POC interrupt flag
POC_INT_FLAG_IN0IF	POC_IN0 input interrupt flag
POC_INT_FLAG_IN1IF	POC_IN1 input interrupt flag
POC_INT_FLAG_IN2IF	POC_IN2 input interrupt flag
POC_INT_FLAG_IN3IF	POC_IN3 input interrupt flag
POC_INT_FLAG_IN4IF	POC_IN4 input interrupt flag
POC_INT_FLAG_IN5IF	POC_IN5 input interrupt flag
POC_INT_FLAG_TIMER0_CCIF	TIMER0 concurrent conduction interrupt flag
POC_INT_FLAG_TIMER7_CCIF	TIMER7 concurrent conduction interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus status;
```

```
/* get POC_IN0 input interrupt flag */
```

```
status = poc_interrupt_flag_get(POC_INT_FLAG_IN0IF);
```

### poc\_interrupt\_flag\_clear

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

**Table 3-722. Function poc\_interrupt\_flag\_clear**

<b>Function name</b>	poc_interrupt_flag_clear
<b>Function prototype</b>	void poc_interrupt_flag_clear(uint32_t int_flag)
<b>Function descriptions</b>	clear POC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	the POC interrupt flag
POC_INT_FLAG_IN0IF	POC_IN0 input interrupt flag
POC_INT_FLAG_IN1IF	POC_IN1 input interrupt flag
POC_INT_FLAG_IN2IF	POC_IN2 input interrupt flag
POC_INT_FLAG_IN3IF	POC_IN3 input interrupt flag
POC_INT_FLAG_IN4IF	POC_IN4 input interrupt flag
POC_INT_FLAG_IN5IF	POC_IN5 input interrupt flag
POC_INT_FLAG_TIMER0_CCIF	TIMER0 concurrent conduction interrupt flag
POC_INT_FLAG_TIMER7_CCIF	TIMER7 concurrent conduction interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear POC_IN0 input interrupt flag */
```

```
poc_interrupt_flag_clear (POC_INT_FLAG_IN0IF);
```

## 3.23. 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 [3.23.1](#), the RCU firmware functions are introduced in chapter [3.23.2](#).

### 3.23.1. Descriptions of Peripheral registers

**Table 3-723. RCU Registers**

Registers	Descriptions
RCU_CTL	control register
RCU_CFG0	clock configuration register 0
RCU_INT	clock interrupt register
RCU_APB2RST	APB2 reset register
RCU_APB1RST	APB1 reset register
RCU_AHBEN	AHB1 enable register
RCU_APB2EN	APB2 enable register
RCU_APB1EN	APB1 enable register
RCU_RSTSCK	reset source / clock register
RCU_AHBRST	AHB reset register
RCU_CFG1	clock configuration register 1
RCU_CFG2	clock configuration register 2

### 3.23.2. Descriptions of Peripheral functions

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

**Table 3-724. 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 peripherals
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_ckout_config	configure the CK_OUT clock source
rcu_pll_config	configure the main PLL clock
rcu_system_reset_enable	enable RCU system reset
rcu_system_reset_disable	disable RCU system reset

rcu_adc_clock_config	configure the CK_ADCPRE clock source and division factor
rcu_i2c_clock_source_config	configure the CK_I2C clock source
rcu_osc_i_stab_wait	wait for oscillator stabilization flags is SET or oscillator startup is timeout
rcu_osc_i_on	turn on the oscillator
rcu_osc_i_off	turn off the oscillator
rcu_osc_i_bypass_mode_enable	enable the oscillator bypass mode, HXTALEN must be reset before it
rcu_osc_i_bypass_mode_disable	disable the oscillator bypass mode, HXTALEN or LXTALEN must be reset before it
rcu_hxtal_frequency_scale_select	HXTAL frequency scale select
rcu_irc32m_adjust_value_set	set the IRC32M adjust value
rcu_hxtal_clock_monitor_enable	enable the HXTAL clock monitor
rcu_hxtal_clock_monitor_disable	disable the HXTAL clock monitor
rcu_pll_clock_monitor_enable	enable the PLL clock monitor
rcu_pll_clock_monitor_disable	disable the PLL clock monitor
rcu_clock_freq_get	get the system clock, bus and peripheral clock frequency
rcu_flag_get	get the clock stabilization and peripheral reset flags
rcu_all_reset_flag_clear	clear all the reset flag
rcu_interrupt_flag_get	get the clock stabilization interrupt and ckm flags
rcu_interrupt_flag_clear	clear the interrupt flags
rcu_interrupt_enable	enable the stabilization interrupt
rcu_interrupt_disable	disable the stabilization interrupt

## Enum rcu\_periph\_enum

**Table 3-725. Enum rcu\_periph\_enum**

Member name	Descriptions
RCU_DMA0	DMA0 clock
RCU_DMA1	DMA1 clock
RCU_DMAMUX	DMAMUX clock
RCU_CRC	CRC 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_GPION	GPION clock
RCU_POC	POC clock
RCU_GTOC	GTOC clock
RCU_SVPWM	SVPWM clock
RCU_TMU	TMU clock
RCU_SYSCFG	SYSCFG clock
RCU_CMP	CMP clock
RCU_ADC0	ADC0 clock
RCU_ADC2	ADC2 clock
RCU_TIMER0	TIMER0 clock
RCU_SPI	SPI clock
RCU_TIMER7	TIMER7 clock
RCU_TIMER1	TIMER1 clock
RCU_TIMER2	TIMER2 clock
RCU_GPTIMER0	GPTIMER0 clock
RCU_GPTIMER1	GPTIMER1 clock
RCU_EVIC	EVIC clock
RCU_CAN	CAN clock
RCU_CPTIMER0	CPTIMER0 clock
RCU_CPTIMER1	CPTIMER1 clock
RCU_CPTIMERW	CPTIMERW clock
RCU_WWDGT	WWDGT clock
RCU_UART0	UART0 clock
RCU_UART1	UART1 clock
RCU_UART2	UART2 clock
RCU_UART3	UART3 clock
RCU_I2C	I2C clock
RCU_CFMU	CFMU clock
RCU_PMU	PMU clock
RCU_DAC	DAC clock

## Enum rcu\_periph\_reset\_enum

**Table 3-726. Enum rcu\_periph\_reset\_enum**

Member name	Descriptions
RCU_DMA0RST	DMA0 clock reset
RCU_DMA1RST	DMA1 clock reset
RCU_DMAMUXRST	DMAMUX clock reset
RCU_CRCRST	CRC clock reset
RCU_GPIOARST	GPIOA clock reset
RCU_GPIOBRST	GPIOB clock reset
RCU_GPIOCRST	GPIOC clock reset

RCU_GPIODRST	GPIOD clock reset
RCU_GPIOERST	GPIOE clock reset
RCU_GPIOFRST	GPIOF clock reset
RCU_GPIOGRST	GPIOG clock reset
RCU_GPIONRST	GPION clock reset
RCU_POCRST	POC clock reset
RCU_GTOCRST	GTOC clock reset
RCU_SVPWMRST	SVPWM clock reset
RCU_TMURST	TMU clock reset
RCU_SYSCFGRST	system configuration reset
RCU_CMPRST	Comparator reset
RCU_ADC0RST	ADC0 clock reset
RCU_ADC2RST	ADC2 clock reset
RCU_TIMER0RST	TIMER0 clock reset
RCU_SPIRST	SPI clock reset
RCU_TIMER7RST	TIMER7 clock reset
RCU_TIMER1RST	TIMER1 clock reset
RCU_TIMER2RST	TIMER2 clock reset
RCU_GPTIMER0RST	GPTIMER0 clock reset
RCU_GPTIMER1RST	GPTIMER1 clock reset
RCU_CANRST	CAN clock reset
RCU_CPTIMER0RST	CPTIMER0 clock reset
RCU_CPTIMER1RST	CPTIMER1 clock reset
RCU_CPTIMERWRST	CPTIMERW clock reset
RCU_WWDGTRST	WWDGT clock reset
RCU_UART0RST	UART0 clock reset
RCU_UART1RST	UART1 clock reset
RCU_UART2RST	UART2 clock reset
RCU_UART3RST	UART3 clock reset
RCU_I2CRST	I2C clock reset
RCU_CFMURST	CFMU clock reset
RCU_PMURST	PMU clock reset
RCU_DACRST	DAC clock reset

### Enum rcu\_periph\_sleep\_enum

**Table 3-727. Enum rcu\_periph\_sleep\_enum**

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

## Enum rcu\_flag\_enum

**Table 3-728. Enum rcu\_flag\_enum**

Member name	Descriptions
RCU_FLAG_IRC32MS TB	IRC32M stabilization flag
RCU_FLAG_HXTALST B	HXTAL stabilization flag
RCU_FLAG_PLLSTB	PLL stabilization flag
RCU_FLAG_IRC32KST B	IRC32K stabilization flag
RCU_FLAG_LVD0RST	low voltage0 detect error reset flag
RCU_FLAG_LOCKUPR ST	CPU LOCK UP error reset flag
RCU_FLAG_LVD1RST	low voltage1 detect error reset flag
RCU_FLAG_LVD2RST	low voltage2 detect error reset flag
RCU_FLAG_LOHRST	lost of HXTAL error reset flag
RCU_FLAG_LOPRST	lost of PLL error reset flag
RCU_FLAG_V11RST	1.1V domain Power reset flag
RCU_FLAG_RSTFC	reset flag clear
RCU_FLAG_OBLRST	option byte loader reset flag
RCU_FLAG_EPRST	external PIN reset flag
RCU_FLAG_PORRST	power reset flag
RCU_FLAG_SWRST	software reset flag
RCU_FLAG_FWDGTR ST	FWDGT reset flag
RCU_FLAG_WWDGTR ST	WWDGT reset flag
RCU_FLAG_LPRST	low-power reset flag

## Enum rcu\_int\_flag\_enum

**Table 3-729. Enum rcu\_int\_flag\_enum**

Member name	Descriptions
RCU_INT_FLAG_IRC3 2KSTB	IRC32K stabilization interrupt flag
RCU_INT_FLAG_IRC3 2MSTB	IRC32M stabilization interrupt flag
RCU_INT_FLAG_HXTA LSTB	HXTAL stabilization interrupt flag
RCU_INT_FLAG_PLLS TB	PLL stabilization interrupt flag
RCU_INT_FLAG_PLLM	PLL clock monitor interrupt flag
RCU_INT_FLAG_CKM	HXTAL clock stuck interrupt flag

## Enum rcu\_int\_flag\_clear\_enum

**Table 3-730. Enum rcu\_int\_flag\_clear\_enum**

Member name	Descriptions
RCU_INT_FLAG_IRC32KSTB_CLR	IRC32K stabilization interrupt flag clear
RCU_INT_FLAG_IRC32MSTB_CLR	IRC32M stabilization interrupt flag clear
RCU_INT_FLAG_HXTALSTB_CLR	HXTAL stabilization interrupt flag clear
RCU_INT_FLAG_PLLSTB_CLR	PLL stabilization interrupt flag clear
RCU_INT_FLAG_PLLM_CLR	PLL clock monitor interrupt clear
RCU_INT_FLAG_CKM_CLR	CKM interrupt flag clear

## Enum rcu\_int\_enum

**Table 3-731. Enum rcu\_int\_enum**

Member name	Descriptions
RCU_INT_IRC32KSTB	IRC32K stabilization interrupt enable
RCU_INT_IRC32MSTB	IRC32M stabilization interrupt enable
RCU_INT_HXTALSTB	HXTAL stabilization interrupt enable
RCU_INT_PLLSTB	PLL stabilization interrupt enable
RCU_INT_PLLM	PLL clock monitor interrupt enable

## Enum rcu\_osci\_type\_enum

**Table 3-732. Enum rcu\_osci\_type\_enum**

Member name	Descriptions
RCU_HXTAL	HXTAL
RCU_IRC32M	IRC32M
RCU_IRC32K	IRC32K
RCU_PLL_CK	PLL

## Enum rcu\_clock\_freq\_enum

**Table 3-733. 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

## rcu\_deinit

The description of rcu\_deinit is shown as below:

**Table 3-734. 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 3-735. 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 3-725. Enum rcu_periph_enum</a>
<i>RCU_TMU</i>	TMU clock
<i>RCU_SVPWM</i>	SVPWM clock
<i>RCU_GTOC</i>	GTOC clock
<i>RCU_POC</i>	POC clock
<i>RCU_GPIOx</i> ( <i>x</i> = <i>A, B, C, D, E, F, G, N</i> )	GPIO ports clock
<i>RCU_CRC</i>	CRC clock
<i>RCU_DMAMUX</i>	DMAMUX clock
<i>RCU_DMAx</i> ( <i>x</i> = <i>0, 1</i> )	DMA clock
<i>RCU_CAN</i>	CAN clock
<i>RCU_EVIC</i>	EVIC clock

<i>RCU_TIMERx</i> ( <i>x</i> = 0,1,2,7)	TIMER clock
<i>RCU_GPTIMERx</i> ( <i>x</i> = 0,1)	GPTIMER clock
<i>RCU_CPTIMERx</i> ( <i>x</i> = 0,1,W)	CPTIMER clock
<i>RCU_SPI</i>	SPI clock
<i>RCU_WWDGT</i>	WWDGT clock
<i>RCU_UARTx</i> ( <i>x</i> = 0,1,2,3)	UART clock
<i>RCU_I2C</i>	I2C clock
<i>RCU_SYSCFG</i>	System configuration clock
<i>RCU_PMU</i>	PMU clock
<i>RCU_CFMU</i>	CFMU clock
<i>RCU_DAC</i>	DAC clock
<i>RCU_ADCx</i> ( <i>x</i> = 0,2)	ADC clock
<i>RCU_CMP</i>	CMP clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the USART0 clock */
```

```
rcu_periph_clock_enable(RCU_USART0);
```

### rcu\_periph\_clock\_disable

The description of rcu\_periph\_clock\_disable is shown as below:

**Table 3-736. 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>	-
Input parameter{in}	
<b>periph</b>	RCU peripherals, refer to <a href="#">Table 3-725. Enum rcu_periph_enum</a>
<i>RCU_TMU</i>	TMU clock
<i>RCU_SVPWM</i>	SVPWM clock
<i>RCU_GTOC</i>	GTOC clock
<i>RCU_POC</i>	POC clock
<i>RCU_GPIOx</i> ( <i>x</i> = A,B,C,D,E,F,G,N)	GPIO ports clock

<i>RCU_CRC</i>	CRC clock
<i>RCU_DMAMUX</i>	DMAMUX clock
<i>RCU_DMAx</i> ( <i>x</i> = 0,1)	DMA clock
<i>RCU_CAN</i>	CAN clock
<i>RCU_EVIC</i>	EVIC clock
<i>RCU_TIMERx</i> ( <i>x</i> = 0,1,2,7)	TIMER clock
<i>RCU_GPTIMERx</i> ( <i>x</i> = 0,1)	GPTIMER clock
<i>RCU_CPTIMERx</i> ( <i>x</i> = 0,1,W)	CPTIMER clock
<i>RCU_SPI</i>	SPI clock
<i>RCU_WWDGT</i>	WWDGT clock
<i>RCU_UARTx</i> ( <i>x</i> =0,1,2,3)	UART clock
<i>RCU_I2C</i>	I2C clock
<i>RCU_SYSCFG</i>	System configuration clock
<i>RCU_PMU</i>	PMU clock
<i>RCU_CFMU</i>	CFMU clock
<i>RCU_DAC</i>	DAC clock
<i>RCU_ADCx</i> ( <i>x</i> = 0,2)	ADC clock
<i>RCU_CMP</i>	CMP clock
<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 3-737. 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 sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>periph</b>	RCU peripherals, refer to <a href="#">Table 3-727. Enum rcu_periph_sleep_enum</a>
<i>RCU_FMC_SLP</i>	FMC clock when sleep mode

<i>RCU_SRAM_SLP</i>	SRAM clock when sleep mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-738. Function rcu\_periph\_clock\_sleep\_disable**

<b>Function name</b>	rcu_periph_clock_sleep_disable
<b>Function prototype</b>	void rcu_periph_clock_sleep_disable(rcu_periph_sleep_enum periph)
<b>Function descriptions</b>	disable the peripherals clock when sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>periph</b>	RCU peripherals, refer to <a href="#">Table 3-727. Enum rcu_periph_sleep_enum</a>
<i>RCU_SRAM_SLP</i>	SRAM clock when sleep mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the SRAM clock when in sleep mode */
rcu_periph_clock_sleep_disable(RCU_SRAM_SLP);
```

### rcu\_periph\_reset\_enable

The description of rcu\_periph\_reset\_enable is shown as below:

**Table 3-739. Function rcu\_periph\_reset\_enable**

<b>Function name</b>	rcu_periph_reset_enable
<b>Function prototype</b>	void rcu_periph_reset_enable(rcu_periph_reset_enum periph_reset)
<b>Function descriptions</b>	reset the peripherals
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>periph_reset</b>	RCU peripherals reset, refer to <a href="#">Table 3-726. Enum</a>



	<a href="#"><u>rcu_periph_reset_enum</u></a>
<i>RCU_TMURST</i>	TMU clock
<i>RCU_SVPWMRST</i>	SVPWM clock
<i>RCU_GTOCRST</i>	GTOC clock
<i>RCU_POCRST</i>	POC clock
<i>RCU_GPIOxRST</i> (x = A,B,C,D,E,F,G,N)	GPIO ports clock
<i>RCU_CRCRST</i>	CRC clock
<i>RCU_DMAMUXRST</i>	DMAMUX clock
<i>RCU_DMAxRST</i> (x = 0,1)	DMA clock
<i>RCU_CANRST</i>	CAN clock
<i>RCU_TIMERxRST</i> (x = 0,1,2,7)	TIMER clock
<i>RCU_GPTIMERxRST</i> (x = 0,1)	GPTIMER clock
<i>RCU_CPTIMERxRST</i> (x = 0,1,W)	CPTIMER clock
<i>RCU_SPIRST</i>	SPI clock
<i>RCU_WWDGTRST</i>	WWDGT clock
<i>RCU_UARTxRST</i> (x =0,1,2,3)	UART clock
<i>RCU_I2CRST</i>	I2C clock
<i>RCU_SYSCFGRST</i>	System configuration clock
<i>RCU_PMURST</i>	PMU clock
<i>RCU_CFMURST</i>	CFMU clock
<i>RCU_DACRST</i>	DAC clock
<i>RCU_ADCxRST</i> (x = 0,2)	ADC clock
<i>RCU_CMPRST</i>	CMP clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI reset */
```

```
rcu_periph_reset_enable(RCU_SPIRST);
```

### **rcu\_periph\_reset\_disable**

The description of rcu\_periph\_reset\_disable is shown as below:

Table 3-740. Function rcu\_periph\_reset\_disable

<b>Function name</b>	rcu_periph_reset_disable
<b>Function prototype</b>	void rcu_periph_reset_disable(rcu_periph_reset_enum periph_reset)
<b>Function descriptions</b>	disable reset the peripherals
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>periph_reset</b>	RCU peripherals reset, refer to <a href="#">Table 3-726. Enum rcu_periph_reset_enum</a>
<i>RCU_TMURST</i>	TMU clock
<i>RCU_SVPWMRST</i>	SVPWM clock
<i>RCU_GTOCRST</i>	GTOC clock
<i>RCU_POCRST</i>	POC clock
<i>RCU_GPIOxRST</i> (x = A,B,C,D,E,F,G,N)	GPIO ports clock
<i>RCU_CRCRST</i>	CRC clock
<i>RCU_DMAMUXRST</i>	DMAMUX clock
<i>RCU_DMAxRST</i> (x = 0,1)	DMA clock
<i>RCU_CANRST</i>	CAN clock
<i>RCU_TIMERxRST</i> (x = 0,1,2,7)	TIMER clock
<i>RCU_GPTIMERxRST</i> (x = 0,1)	GPTIMER clock
<i>RCU_CPTIMERxRST</i> (x = 0,1,W)	CPTIMER clock
<i>RCU_SPIRST</i>	SPI clock
<i>RCU_WWDGTRST</i>	WWDGT clock
<i>RCU_UARTxRST</i> (x = 0,1,2,3)	UART clock
<i>RCU_I2CRST</i>	I2C clock
<i>RCU_SYSCFGRST</i>	System configuration clock
<i>RCU_PMURST</i>	PMU clock
<i>RCU_CFMURST</i>	CFMU clock
<i>RCU_DACRST</i>	DAC clock
<i>RCU_ADCxRST</i> (x = 0,2)	ADC clock
<i>RCU_CMPRST</i>	CMP clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI reset */
```

```
rcu_periph_reset_disable(RCU_SPIRST);
```

### rcu\_system\_clock\_source\_config

The description of rcu\_system\_clock\_source\_config is shown as below:

**Table 3-741. Function rcu\_system\_clock\_source\_config**

<b>Function name</b>	rcu_system_clock_source_config
<b>Function prototype</b>	void rcu_system_clock_source_config(uint32_t ck_sys)
<b>Function descriptions</b>	configure the system clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_sys</b>	system clock source select
<i>RCU_CKSYSSRC_IRC32M</i>	select CK_IRC32M as the CK_SYS source
<i>RCU_CKSYSSRC_HXTAL</i>	select CK_HXTAL as the CK_SYS source
<i>RCU_CKSYSSRC_PLL</i>	select CK_PLL as the CK_SYS source
<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 3-742. 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>	which clock is selected as CK_SYS source
<i>RCU_SCSS_IRC32M</i>	CK_IRC32M is selected as the CK_SYS source
<i>RCU_SCSS_HXTAL</i>	CK_HXTAL is selected as the CK_SYS source
<i>RCU_SCSS_PLL</i>	CK_PLL is selected as the CK_SYS source

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 3-743. 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_DIV1</i> Vx (x = 1, 2, 4, 8, 16, 64, 128, 256, 512)	select CK_SYS / x as CK_AHB
<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 3-744. 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>	-

Input parameter{in}	
<b>ck_apb1</b>	APB1 clock prescaler selection
<i>RCU_APB1_CKAHB_D</i> <i>IV1</i>	select CK_AHB as CK_APB1
<i>RCU_APB1_CKAHB_D</i> <i>IV2</i>	select CK_AHB / 2 as CK_APB1
<i>RCU_APB1_CKAHB_D</i> <i>IV4</i>	select CK_AHB / 4 as CK_APB1
<i>RCU_APB1_CKAHB_D</i> <i>IV8</i>	select CK_AHB / 8 as CK_APB1
<i>RCU_APB1_CKAHB_D</i> <i>IV16</i>	select CK_AHB / 16 as CK_APB1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/16 as CK_APB1 */
```

```
rcu_apb1_clock_config(RCU_APB1_CKAHB_DIV16);
```

### rcu\_apb2\_clock\_config

The description of rcu\_apb2\_clock\_config is shown as below:

**Table 3-745. 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>	-
Input parameter{in}	
<b>ck_apb2</b>	APB2 clock prescaler selection
<i>RCU_APB2_CKAHB_D</i> <i>IV1</i>	select CK_AHB as CK_APB2
<i>RCU_APB2_CKAHB_D</i> <i>IV2</i>	select CK_AHB/2 as CK_APB2
<i>RCU_APB2_CKAHB_D</i> <i>IV4</i>	select CK_AHB/4 as CK_APB2
<i>RCU_APB2_CKAHB_D</i> <i>IV8</i>	select CK_AHB/8 as CK_APB2
<i>RCU_APB2_CKAHB_D</i> <i>IV16</i>	select CK_AHB/16 as CK_APB2
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB2 */
```

```
rcu_apb2_clock_config(RCU_APB2_CKAHB_DIV8);
```

### rcu\_ckout\_config

The description of rcu\_ckout\_config is shown as below:

**Table 3-746. 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
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ckout_src</b>	CK_OUT clock source selection
<i>RCU_CKOUTSRC_NONE</i>	no clock selected
<i>RCU_CKOUTSRC_IRC32K</i>	<i>IRC32K selected</i>
<i>RCU_CKOUTSRC_CKSYS</i>	system clock selected
<i>RCU_CKOUTSRC_IRC32M</i>	high speed 8M internal oscillator clock selected
<i>RCU_CKOUTSRC_HXTAL</i>	HXTAL selected
<i>RCU_CKOUTSRC_CKPLL_DIV1</i>	CK_PLL selected
<i>RCU_CKOUTSRC_CKPLL_DIV8</i>	CK_PLL/8 selected
<b>Input parameter{in}</b>	
<b>ckout_div</b>	CK_OUT divider
<i>RCU_CKOUT_DIVx(x=1,2,4,8,16,32,64,128)</i>	CK_OUT is divided by x
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the HXTAL as CK_OUT clock source */
```

```
rcu_ckout_config(RCU_CKOUTSRC_HXTAL, RCU_CKOUT_DIV1);
```

### rcu\_pll\_config

The description of rcu\_pll\_config is shown as below:

**Table 3-747. Function rcu\_pll\_config**

Function name	rcu_pll_config
Function prototype	void rcu_pll_config(uint32_t pll_src, uint32_t prediv_div, uint32_t pll_mul)
Function descriptions	configure the main PLL clock
Precondition	-
The called functions	-
Input parameter{in}	
pll_src	PLL clock source selection
RCU_PLLSRC_IRC32M	IRC32M clock selected as source clock of PLL
RCU_PLLSRC_HXTAL	HXTAL selected as source clock of PLL
Input parameter{in}	
prediv_div	PREDV division factor
RCU_PREDV_DIVx	x = 1..8
Input parameter{in}	
pll_mul	PLL clock multiplication factor
RCU_PLL_MULx (x=4,4.5,5,5.5...63.5,64)	PLL clock * x
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the PLL */
```

```
rcu_pll_config(RCU_PLLSRC_HXTAL, RCU_PREDV_DIV1, RCU_PLL_MUL10);
```

### rcu\_system\_reset\_enable

The description of rcu\_system\_reset\_enable is shown as below:

**Table 3-748. Function rcu\_system\_reset\_enable**

Function name	rcu_system_reset_enable
Function prototype	void rcu_system_reset_enable(uint32_t reset_source)
Function descriptions	enable RCU system reset
Precondition	-
The called functions	-

Input parameter{in}	
<b>reset_source</b>	reset source
<i>RCU_SYSRST_LOCKUP</i>	CPU lock-up reset
<i>RCU_SYSRST_LOH</i>	lost of HXTAL reset
<i>RCU_SYSRST_LOP</i>	lost of PLL reset
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RCU system reset */
```

```
rcu_system_reset_enable(RCU_SYSRST_LOCKUP);
```

### rcu\_system\_reset\_disable

The description of rcu\_system\_reset\_disable is shown as below:

**Table 3-749. Function rcu\_system\_reset\_disable**

<b>Function name</b>	rcu_system_reset_disable
<b>Function prototype</b>	void rcu_system_reset_disable(uint32_t reset_source)
<b>Function descriptions</b>	disable RCU system reset
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>reset_source</b>	reset source
<i>RCU_SYSRST_LOCKUP</i>	CPU lock-up reset
<i>RCU_SYSRST_LOH</i>	lost of HXTAL reset
<i>RCU_SYSRST_LOP</i>	lost of PLL reset
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable RCU system reset */
```

```
rcu_system_reset_disable(RCU_SYSRST_LOCKUP);
```

### rcu\_adc\_clock\_config

The description of rcu\_adc\_clock\_config is shown as below:



Table 3-750. Function rcu\_adc\_clock\_config

<b>Function name</b>	rcu_adc_clock_config
<b>Function prototype</b>	void rcu_adc_clock_config(uint32_t ck_adc, uint32_t adc_psc)
<b>Function descriptions</b>	configure the CK_ADCPRE clock source and division factor
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_adc</b>	ADC clock source select
<i>RCU_CK_ADCPRE_PCLK2</i>	select CK_PCLK2 as the CK_ADC source
<i>RCU_CK_ADCPRE_HXTAL</i>	select CK_HXTAL as the CK_ADC source
<i>RCU_CK_ADCPRE_PLL</i>	select CK_PLL as the CK_ADC source
<i>RCU_CK_ADCPRE_IRC32M</i>	select IRC32M as the CK_ADC source
<b>Input parameter{in}</b>	
<b>adc_psc</b>	ADC prescaler factor
<i>RCU_CK_ADCPRE_DIVx(x = 2,3...31,32)</i>	ADC prescaler select CK_ADCPRE/ x
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the ADC prescaler factor */
```

```
rcu_adc_clock_config(RCU_CK_ADCPRE_PLL, RCU_CK_ADCPRE_DIV2);
```

### rcu\_i2c\_clock\_source\_config

The description of rcu\_i2c\_clock\_source\_config is shown as below:

Table 3-751. Function rcu\_i2c\_clock\_source\_config

<b>Function name</b>	rcu_i2c_clock_source_config
<b>Function prototype</b>	void rcu_i2c_clock_source_config(uint32_t ck_i2c)
<b>Function descriptions</b>	configure the CK_I2C clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_i2c</b>	i2c clock source select
<i>RCU_CK_I2CSRC_PCLK1</i>	select CK_PCLK1 as the CK_I2C source
<i>RCU_CK_I2CSRC_PLL</i>	select CK_PLL as the CK_I2C source

<i>RCU_CK_I2CSRC_IRC32M</i>	select CK_IRC32M as the CK_I2C source
<i>RCU_CK_I2CSRC_HXTAL</i>	select CK_HXTAL as the CK_I2C source
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK_I2C clock source */
rcu_i2c_clock_source_config(RCU_CK_I2CSRC_PLL);
```

### rcu\_osc\_i\_stab\_wait

The description of rcu\_osc\_i\_stab\_wait is shown as below:

**Table 3-752. 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_i_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 3-732. Enum rcu_osc_i_type_enum</a>
<i>RCU_HXTAL</i>	high speed crystal oscillator(HXTAL)
<i>RCU_IRC32M</i>	internal 32M RC oscillators(IRC32M)
<i>RCU_IRC32K</i>	internal 32K RC oscillator(IRC32K)
<i>RCU_PLL_CK</i>	phase locked loop(PLL)
<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 3-753. Function rcu\_osci\_on

Function name	rcu_osci_on
Function prototype	void rcu_osci_on(rcu_osci_type_enum osci)
Function descriptions	turn on the oscillator
Precondition	-
The called functions	-
Input parameter{in}	
osci	oscillator types, refer to rcu_osci_type_enum
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_IRC32M	internal 32M RC oscillators(IRC32M)
RCU_IRC32K	internal 32K RC oscillator(IRC32K)
RCU_PLL_CK	phase locked loop(PLL)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn on the high speed crystal oscillator */
```

```
rcu_osci_on(RCU_HXTAL);
```

### rcu\_osci\_off

The description of rcu\_osci\_off is shown as below:

Table 3-754. 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 rcu_osci_type_enum
RCU_HXTAL	high speed crystal oscillator(HXTAL)
RCU_IRC32M	internal 32M RC oscillators(IRC32M)
RCU_IRC32K	internal 32K RC oscillator(IRC32K)
RCU_PLL_CK	phase locked loop(PLL)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn off the high speed crystal oscillator */
```

```
rcu_osci_off(RCU_HXTAL);
```

### rcu\_osci\_bypass\_mode\_enable

The description of rcu\_osci\_bypass\_mode\_enable is shown as below:

**Table 3-755. Function rcu\_osci\_bypass\_mode\_enable**

<b>Function name</b>	rcu_osci_bypass_mode_enable
<b>Function prototype</b>	void rcu_osci_bypass_mode_enable()
<b>Function descriptions</b>	enable the oscillator bypass mode, HXTALEN must be reset before it
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the high speed crystal oscillator bypass mode */
```

```
rcu_osci_bypass_mode_enable();
```

### rcu\_osci\_bypass\_mode\_disable

The description of rcu\_osci\_bypass\_mode\_disable is shown as below:

**Table 3-756. 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(void)
<b>Function descriptions</b>	disable the oscillator bypass mode, HXTALEN must be reset before it
<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 high speed crystal oscillator bypass mode */
```

```
rcu_osci_bypass_mode_disable();
```

### rcu\_hxtal\_frequency\_scale\_select

The description of rcu\_hxtal\_frequency\_scale\_select is shown as below:

Table 3-757. Function rcu\_hxtal\_frequency\_scale\_select

<b>Function name</b>	rcu_hxtal_frequency_scale_select
<b>Function prototype</b>	void rcu_hxtal_frequency_scale_select(uint32_t hxtal_scal)
<b>Function descriptions</b>	HXTAL frequency scale select
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>hxtal_scal</b>	HXTAL frequency scale
<i>HXTAL_SCALE_1M_TO_10M</i>	HXTAL scale is 1-10MHz
<i>HXTAL_SCALE_10M_TO_24M</i>	HXTAL scale is 10-24MHz
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* HXTAL frequency scale select */
```

```
rcu_hxtal_frequency_scale_select(HXTAL_SCALE_1M_TO_10M);
```

### rcu\_irc32m\_adjust\_value\_set

The description of rcu\_irc32m\_adjust\_value\_set is shown as below:

Table 3-758. Function rcu\_irc32m\_adjust\_value\_set

<b>Function name</b>	rcu_irc32m_adjust_value_set
<b>Function prototype</b>	void rcu_irc32m_adjust_value_set(uint32_t irc32m_adjval)
<b>Function descriptions</b>	set the IRC32M adjust value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>irc32m_adjval</b>	IRC32M adjust value, must be between 0 and 0x1F
<i>0x00 - 0x1F</i>	IRC32M adjust value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the IRC32M adjust value */
```

```
rcu_irc32m_adjust_value_set(0x10);
```

## rcu\_hxtal\_clock\_monitor\_enable

The description of rcu\_hxtal\_clock\_monitor\_enable is shown as below:

**Table 3-759. 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 3-760. 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>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## rcu\_pll\_clock\_monitor\_enable

The description of rcu\_pll\_clock\_monitor\_enable is shown as below:

**Table 3-761. Function rcu\_pll\_clock\_monitor\_enable**

<b>Function name</b>	rcu_pll_clock_monitor_enable
<b>Function prototype</b>	void rcu_pll_clock_monitor_enable(void)
<b>Function descriptions</b>	enable the PLL 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 PLL clock monitor */
rcu_pll_clock_monitor_enable();
```

### rcu\_pll\_clock\_monitor\_disable

The description of rcu\_pll\_clock\_monitor\_disable is shown as below:

**Table 3-762. Function rcu\_pll\_clock\_monitor\_disable**

<b>Function name</b>	rcu_pll_clock_monitor_disable
<b>Function prototype</b>	void rcu_pll_clock_monitor_disable(void)
<b>Function descriptions</b>	disable the PLL 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:

```
/* disable the PLL clock monitor */
rcu_pll_clock_monitor_disable();
```

### rcu\_clock\_freq\_get

The description of rcu\_clock\_freq\_get is shown as below:

**Table 3-763. 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 and peripheral clock frequency
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clock</b>	the clock frequency which to get, refer to <a href="#">Table 3-733. Enum rcu_clock_freq_enum</a>
CK_SYS	system clock frequency
CK_AHB	AHB clock frequency
CK_APB1	APB1 clock frequency
CK_APB2	APB2 clock frequency
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ck_freq</b>	clock frequency of system, AHB, APB1, APB2

Example:

```
uint32_t temp_freq;

/* get the system clock frequency */

temp_freq = rcu_clock_freq_get(CK_SYS);
```

### rcu\_flag\_get

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

**Table 3-764. 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>	
<b>flag</b>	the clock stabilization and peripheral reset flags, refer to <a href="#">Table 3-728. Enum rcu_flag_enum</a>
RCU_FLAG_IRC32MS TB	IRC32M stabilization flag
RCU_FLAG_HXTALST B	HXTAL stabilization flag
RCU_FLAG_PLLSTB	PLL stabilization flag
RCU_FLAG_IRC32KST B	IRC32K stabilization flag
RCU_FLAG_LVD0RST	Low Voltage 0 detect error reset flag
RCU_FLAG_LOCKUPR	CPU LOCK UP Error reset flag



<i>ST</i>	
<i>RCU_FLAG_LVD1RST</i>	Low Voltage 1 Detect Error reset flag
<i>RCU_FLAG_LVD2RST</i>	Low Voltage 2 Detect Error reset flag
<i>RCU_FLAG_LOHRST</i>	lost of HXTAL Error reset flag
<i>RCU_FLAG_LOPRST</i>	lost of PLL Error reset flag
<i>RCU_FLAG_V11RST</i>	1.1V domain Power reset flag
<i>RCU_FLAG_OBLRST</i>	option byte loader reset flag
<i>RCU_FLAG_EPRST</i>	external PIN reset flag
<i>RCU_FLAG_PORRST</i>	power reset flag
<i>RCU_FLAG_SWRST</i>	software reset flag
<i>RCU_FLAG_FWDGTR</i> <i>ST</i>	free watchdog timer reset flag
<i>RCU_FLAG_WWDGTR</i> <i>ST</i>	window watchdog timer reset flag
<i>RCU_FLAG_LPRST</i>	low-power reset flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### rcu\_all\_reset\_flag\_clear

The description of rcu\_all\_reset\_flag\_clear is shown as below:

**Table 3-765. Function rcu\_all\_reset\_flag\_clear**

<b>Function name</b>	rcu_all_reset_flag_clear
<b>Function prototype</b>	void rcu_all_reset_flag_clear(void)
<b>Function descriptions</b>	clear all the reset flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear all the reset flag */
```

```
rcu_all_reset_flag_clear();
```

### rcu\_interrupt\_flag\_get

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

**Table 3-766. 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, refer to <a href="#">Table 3-729. Enum rcu_int_flag_enum</a>
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	interrupt and ckm flags, refer to rcu_int_flag_enum
RCU_INT_FLAG_IRC3 2KSTB	IRC32K stabilization interrupt flag
RCU_INT_FLAG_IRC3 2MSTB	IRC32M 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_PLLM	PLL clock monitor interrupt flag
<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 3-767. 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 3-730. Enum rcu_int_flag_clear_enum</a>
<i>RCU_INT_FLAG_IRC32KSTB_CLR</i>	IRC32K stabilization interrupt flag clear
<i>RCU_INT_FLAG_IRC32MSTB_CLR</i>	IRC32M stabilization interrupt flag clear
<i>RCU_INT_FLAG_HXTALSTB_CLR</i>	HXTAL stabilization interrupt flag clear
<i>RCU_INT_FLAG_PLLSTB_CLR</i>	PLL stabilization interrupt flag clear
<i>RCU_INT_FLAG_CKM_CLR</i>	clock stuck interrupt flag clear
<i>RCU_INT_FLAG_PLLM_CLR</i>	PLL clock monitor interrupt clear
<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);
```

### rcu\_interrupt\_enable

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

**Table 3-768. Function rcu\_interrupt\_enable**

<b>Function name</b>	rcu_interrupt_enable
<b>Function prototype</b>	void rcu_interrupt_enable(rcu_int_enum interrupt)
<b>Function descriptions</b>	enable the stabilization interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	clock stabilization interrupt, refer to <a href="#">Table 3-731. Enum rcu_int_enum</a>
<i>RCU_INT_IRC32KSTB</i>	IRC32K stabilization interrupt enable
<i>RCU_INT_IRC32MSTB</i>	IRC32M stabilization interrupt enable
<i>RCU_INT_HXTALSTB</i>	HXTAL stabilization interrupt enable
<i>RCU_INT_PLLSTB</i>	PLL stabilization interrupt enable
<i>RCU_INT_PLLM</i>	PLL clock monitor interrupt enable
<b>Output parameter{out}</b>	

-	-
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 3-769. 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>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	clock stabilization interrupt, refer to <a href="#">Table 3-731. Enum rcu_int_enum</a>
<i>RCU_INT_IRC32KSTB</i>	IRC32K stabilization interrupt disable
<i>RCU_INT_IRC32MSTB</i>	IRC32M stabilization interrupt disable
<i>RCU_INT_HXTALSTB</i>	HXTAL stabilization interrupt disable
<i>RCU_INT_PLLSTB</i>	PLL stabilization interrupt disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## 3.24. SPI

The SPI module can communicate with external devices using the SPI protocol. The SPI registers are listed in chapter [3.24.1](#), the SPI firmware functions are introduced in chapter [3.24.2](#).

### 3.24.1. Descriptions of Peripheral registers

SPI registers are listed in the table shown as below:

**Table 3-770. 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

### 3.24.2. Descriptions of Peripheral functions

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

**Table 3-771. SPI firmware function**

Function name	Function description
spi_deinit	reset SPI peripheral
spi_struct_para_init	initialize the parameters of SPI struct with the default values
spi_init	initialize SPI peripheral parameter
spi_enable	enable SPI
spi_disable	disable SPI
spi_nss_output_enable	enable SPI NSS output function
spi_nss_output_disable	disable SPI NSS output function
spi_nss_internal_high	SPI NSS pin high level in software mode
spi_nss_internal_low	SPI NSS pin low level in software mode
spi_dma_enable	enable SPI DMA function
spi_dma_disable	disable SPI DMA function
spi_data_frame_format_config	configure SPI data frame format
spi_data_transmit	SPI transmit data
spi_data_receive	SPI receive data
spi_bidirectional_transfer_config	configure SPI bidirectional transfer direction
spi_crc_polynomial_set	set SPI CRC polynomial
spi_crc_polynomial_get	get SPI CRC polynomial
spi_crc_on	turn on SPI CRC function
spi_crc_off	turn off SPI CRC function
spi_crc_next	SPI next data is CRC value
spi_crc_get	get SPI CRC send value or receive value
spi_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

Function name	Function description
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_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 3-772. spi\_parameter\_struct

Member name	Function description
device_mode	SPI master or slave (SPI_MASTER, SPI_SLAVE)
trans_mode	SPI transtype (SPI_TRANSMODE_FULLDUPLEX, SPI_TRANSMODE_RECEIVEONLY, SPI_TRANSMODE_BDRECEIVE, SPI_TRANSMODE_BDTRANSMIT)
frame_size	SPI frame size (SPI_FRAME_SIZE_16BIT, SPI_FRAME_SIZE_8BIT)
nss	SPI NSS control by hardware or software (SPI_NSS_SOFT, SPI_NSS_HARD)
endian	SPI big endian or little endian (SPI_ENDIAN_MSB, SPI_ENDIAN_LSB)
clock_polarity_phase	SPI clock phase and polarity (SPI_CK_PL_LOW_PH_1EDGE, SPI_CK_PL_HIGH_PH_1EDGE, SPI_CK_PL_LOW_PH_2EDGE, SPI_CK_PL_HIGH_PH_2EDGE)
prescale	SPI prescale factor (SPI_PSC_n (n=2,4,8,16,32,64,128,256))

## spi\_deinit

The description of spi\_deinit is shown as below:

Table 3-773. Function spi\_deinit

Function name	spi_deinit
Function prototype	void spi_deinit(void);
Function descriptions	reset SPI peripheral
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* reset SPI */
```

```
spi_deinit();
```

### spi\_struct\_para\_init

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

**Table 3-774. Function spi\_struct\_para\_init**

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

Example:

```
/* initialize the parameters of SPI */
```

```
spi_parameter_struct spi_init_struct;
```

```
spi_struct_para_init(&spi_init_struct);
```

### spi\_init

The description of spi\_init is shown as below:

**Table 3-775. Function spi\_init**

Function name	spi_init
Function prototype	void spi_init(spi_parameter_struct* spi_struct);
Function descriptions	initialize SPI peripheral parameter
Precondition	-
The called functions	-
Input parameter{in}	
spi_struct	SPI parameter initialization struct, the structure members can refer to members of the structure <a href="#">Table 3-772. spi_parameter_struct</a>
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* initialize SPI */

spi_parameter_struct spi_init_struct;

spi_init_struct.trans_mode      = SPI_TRANSMODE_BDTRANSMIT;
spi_init_struct.device_mode     = SPI_MASTER;
spi_init_struct.frame_size      = SPI_FRAMESIZE_8BIT;
spi_init_struct.clock_polarity_phase = SPI_CK_PL_HIGH_PH_2EDGE;
spi_init_struct.nss             = SPI_NSS_SOFT;
spi_init_struct.prescale        = SPI_PSC_8;
spi_init_struct.endian          = SPI_ENDIAN_MSB;

spi_init(&spi_init_struct);
```

### spi\_enable

The description of spi\_enable is shown as below:

**Table 3-776. Function spi\_enable**

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

Example:

```
/* enable SPI */

spi_enable();
```

### spi\_disable

The description of spi\_disable is shown as below:



**Table 3-777. Function spi\_disable**

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

Example:

```
/* disable SPI */
spi_disable();
```

### spi\_nss\_output\_enable

The description of spi\_nss\_output\_enable is shown as below:

**Table 3-778. Function spi\_nss\_output\_enable**

<b>Function name</b>	spi_nss_output_enable
<b>Function prototype</b>	void spi_nss_output_enable(void);
<b>Function descriptions</b>	enable SPI NSS output 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 SPI NSS output */
spi_nss_output_enable();
```

### spi\_nss\_output\_disable

The description of spi\_nss\_output\_disable is shown as below:

**Table 3-779. Function spi\_nss\_output\_disable**

<b>Function name</b>	spi_nss_output_disable
----------------------	------------------------

<b>Function prototype</b>	void spi_nss_output_disable(void);
<b>Function descriptions</b>	disable SPI NSS output 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 SPI NSS output */
```

```
spi_nss_output_disable();
```

### spi\_nss\_internal\_high

The description of spi\_nss\_internal\_high is shown as below:

**Table 3-780. Function spi\_nss\_internal\_high**

<b>Function name</b>	spi_nss_internal_high
<b>Function prototype</b>	void spi_nss_internal_high(void);
<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>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI NSS pin is pulled high level in software mode */
```

```
spi_nss_internal_high();
```

### spi\_nss\_internal\_low

The description of spi\_nss\_internal\_low is shown as below:

**Table 3-781. Function spi\_nss\_internal\_low**

<b>Function name</b>	spi_nss_internal_low
<b>Function prototype</b>	void spi_nss_internal_low(void);
<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>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI NSS pin is pulled low level in software mode */
```

```
spi_nss_internal_low();
```

### spi\_dma\_enable

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

**Table 3-782. Function spi\_dma\_enable**

<b>Function name</b>	spi_dma_enable
<b>Function prototype</b>	void spi_dma_enable(uint8_t dma);
<b>Function descriptions</b>	enable SPI DMA function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>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 SPI transmit data DMA function */
```

```
spi_dma_enable(SPI_DMA_TRANSMIT);
```

### spi\_dma\_disable

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

**Table 3-783. Function spi\_dma\_disable**

<b>Function name</b>	spi_dma_disable
<b>Function prototype</b>	void spi_dma_disable(uint8_t dma);
<b>Function descriptions</b>	disable SPI DMA function

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>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 SPI transmit data DMA function */
```

```
spi_dma_disable(SPI_DMA_TRANSMIT);
```

### spi\_data\_frame\_format\_config

The description of spi\_data\_frame\_format\_config is shown as below:

**Table 3-784. Function spi\_data\_frame\_format\_config**

<b>Function name</b>	spi_data_frame_format_config
<b>Function prototype</b>	void spi_data_frame_format_config(uint16_t frame_format);
<b>Function descriptions</b>	configure SPI data frame format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>frame_format</b>	SPI frame size
<i>SPI_FRAME_SIZE_16BIT</i>	SPI frame size is 16 bits
<i>SPI_FRAME_SIZE_8BIT</i>	SPI frame size is 8 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure SPI data frame format size is 16 bits */
```

```
spi_data_frame_format_config(SPI_FRAME_SIZE_16BIT);
```

### spi\_data\_transmit

The description of spi\_data\_transmit is shown as below:

Table 3-785. Function spi\_data\_transmit

Function name	spi_data_transmit
Function prototype	void spi_data_transmit(uint16_t data);
Function descriptions	SPI transmit data
Precondition	-
The called functions	-
Input parameter{in}	
data	16-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI transmit data */
spi_data_transmit(SPI_send_array[send_n]);
```

### spi\_data\_receive

The description of spi\_data\_receive is shown as below:

Table 3-786. Function spi\_data\_receive

Function name	spi_data_receive
Function prototype	uint16_t spi_data_receive(void);
Function descriptions	SPI receive data
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit data

Example:

```
/* SPI receive data */
SPI_receive_array[receive_n] = spi_data_receive();
```

### spi\_bidirectional\_transfer\_config

The description of spi\_bidirectional\_transfer\_config is shown as below:

Table 3-787. Function spi\_bidirectional\_transfer\_config

Function name	spi_bidirectional_transfer_config
---------------	-----------------------------------

<b>Function prototype</b>	void spi_bidirectional_transfer_config(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>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:

```
/* SPI works in transmit-only mode */
```

```
spi_bidirectional_transfer_config(SPI_BIDIRECTIONAL_TRANSMIT);
```

### spi\_format\_error\_clear

The description of spi\_format\_error\_clear is shown as below:

**Table 3-788. Function spi\_format\_error\_clear**

<b>Function name</b>	spi_format_error_clear
<b>Function prototype</b>	void spi_format_error_clear(void);
<b>Function descriptions</b>	clear TI Mode Format Error flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear TI Mode Format Error flag status */
```

```
spi_format_error_clear();
```

### spi\_crc\_polynomial\_set

The description of spi\_crc\_polynomial\_set is shown as below:

Table 3-789. Function spi\_crc\_polynomial\_set

Function name	spi_crc_polynomial_set
Function prototype	void spi_crc_polynomial_set(uint16_t crc_poly);
Function descriptions	set SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
crc_poly	CRC polynomial value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set SPI CRC polynomial */
spi_crc_polynomial_set(CRC_VALUE);
```

### spi\_crc\_polynomial\_get

The description of spi\_crc\_polynomial\_get is shown as below:

Table 3-790. Function spi\_crc\_polynomial\_get

Function name	spi_crc_polynomial_get
Function prototype	uint16_t spi_crc_polynomial_get(void);
Function descriptions	get SPI CRC polynomial
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint16_t	16-bit CRC polynomial (0-0xFFFF)

Example:

```
/* get SPI CRC polynomial */
uint16_t crc_val;
crc_val = spi_crc_polynomial_get();
```

### spi\_crc\_on

The description of spi\_crc\_on is shown as below:

Table 3-791. Function spi\_crc\_on

Function name	spi_crc_on
Function prototype	void spi_crc_on(void);
Function descriptions	turn on SPI CRC function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn on SPI CRC function */
spi_crc_on();
```

### spi\_crc\_off

The description of spi\_crc\_off is shown as below:

Table 3-792. Function spi\_crc\_off

Function name	spi_crc_off
Function prototype	void spi_crc_off(void);
Function descriptions	turn off SPI CRC function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn off SPI CRC function */
spi_crc_off();
```

### spi\_crc\_next

The description of spi\_crc\_next is shown as below:

Table 3-793. Function spi\_crc\_next

Function name	spi_crc_next
---------------	--------------



<b>Function prototype</b>	void spi_crc_next(void);
<b>Function descriptions</b>	SPI next data is CRC value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI next data is CRC value */
```

```
spi_crc_next();
```

### spi\_crc\_get

The description of spi\_crc\_get is shown as below:

**Table 3-794. Function spi\_crc\_get**

<b>Function name</b>	spi_crc_get
<b>Function prototype</b>	uint16_t spi_crc_get(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>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>uint16_t</b>	16-bit CRC value (0-0xFFFF)

Example:

```
/* get SPI CRC send value */
```

```
uint16_t crc_val;
```

```
crc_val = spi_crc_get(SPI_CRC_TX);
```

### spi\_crc\_error\_clear

The description of spi\_crc\_error\_clear is shown as below:

Table 3-795. Function spi\_crc\_error\_clear

Function name	spi_crc_error_clear
Function prototype	void spi_crc_error_clear(void);
Function descriptions	clear SPI CRC error flag status
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear SPI CRC error flag status */
```

### spi\_ti\_mode\_enable

The description of spi\_ti\_mode\_enable is shown as below:

Table 3-796. Function spi\_ti\_mode\_enable

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

Example:

```
/* enable SPI TI mode */
```

```
spi_ti_mode_enable();
```

### spi\_ti\_mode\_disable

The description of spi\_ti\_mode\_disable is shown as below:

Table 3-797. Function spi\_ti\_mode\_disable

Function name	spi_ti_mode_disable
Function prototype	void spi_ti_mode_disable(void);
Function descriptions	disable SPI TI 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 SPI TI mode */
```

```
spi_ti_mode_disable();
```

### spi\_nssp\_mode\_enable

The description of spi\_nssp\_mode\_enable is shown as below:

**Table 3-798. Function spi\_nssp\_mode\_enable**

<b>Function name</b>	spi_nssp_mode_enable
<b>Function prototype</b>	void spi_nssp_mode_enable(void);
<b>Function descriptions</b>	enable SPI NSS pulse 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 SPI NSS pulse mode */
```

```
spi_nssp_mode_enable();
```

### spi\_nssp\_mode\_disable

The description of spi\_nssp\_mode\_disable is shown as below:

**Table 3-799. Function spi\_nssp\_mode\_disable**

<b>Function name</b>	spi_nssp_mode_disable
<b>Function prototype</b>	void spi_nssp_mode_disable(void);
<b>Function descriptions</b>	disable SPI NSS pulse mode
<b>Precondition</b>	-
<b>The called functions</b>	-

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

Example:

```
/* disable SPI NSS pulse mode */
spi_nssp_mode_disable();
```

### spi\_quad\_enable

The description of spi\_quad\_enable is shown as below:

**Table 3-800. Function spi\_quad\_enable**

Function name	spi_quad_enable
Function prototype	void spi_quad_enable(void);
Function descriptions	enable quad wire SPI
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI quad wire mode */
spi_quad_enable();
```

### spi\_quad\_disable

The description of spi\_quad\_disable is shown as below:

**Table 3-801. Function spi\_quad\_disable**

Function name	spi_quad_disable
Function prototype	spi_quad_disable(void);
Function descriptions	disable quad wire SPI
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI quad wire mode */
```

```
spi_quad_disable();
```

### spi\_quad\_write\_enable

The description of spi\_quad\_write\_enable is shown as below:

**Table 3-802. Function spi\_quad\_write\_enable**

Function name	spi_quad_write_enable
Function prototype	void spi_quad_write_enable(void);
Function descriptions	enable quad wire SPI write
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable SPI quad wire write */
```

```
spi_quad_write_enable();
```

### spi\_quad\_read\_enable

The description of spi\_quad\_read\_enable is shown as below:

**Table 3-803. Function spi\_quad\_read\_enable**

Function name	spi_quad_read_enable
Function prototype	void spi_quad_read_enable(void);
Function descriptions	enable quad wire SPI read
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* enable SPI quad wire read */
```

```
spi_quad_read_enable();
```

### spi\_flag\_get

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

**Table 3-804. Function spi\_flag\_get**

Function name	spi_flag_get
Function prototype	FlagStatus spi_flag_get(uint32_t flag);
Function descriptions	get SPI flag status
Precondition	-
The called functions	-
Input parameter{in}	
flag	SPI flag status
SPI_FLAG_TBE	transmit buffer empty flag
SPI_FLAG_RBNE	receive buffer not empty flag
SPI_FLAG_TRANS	transmit on-going flag
SPI_INT_FLAG_RXORERR	receive overrun error flag
SPI_FLAG_CONFERR	mode config error flag
SPI_FLAG_CRCERR	CRC error flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get SPI transmit buffer empty flag status */
```

```
while(RESET == spi_flag_get(SPI_FLAG_TBE));
```

```
spi_data_transmit(SPI_send_array[send_n++]);
```

### spi\_interrupt\_enable

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

**Table 3-805. Function spi\_interrupt\_enable**

Function name	spi_interrupt_enable
Function prototype	void spi_interrupt_enable(uint32_t interrupt);
Function descriptions	enable SPI interrupt

<b>Precondition</b>	-
<b>The called functions</b>	-
<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 SPI transmit buffer empty interrupt */
spi_interrupt_enable(SPI_INT_TBE);
```

### **spi\_interrupt\_disable**

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

**Table 3-806. Function spi\_interrupt\_disable**

<b>Function name</b>	spi_interrupt_disable
<b>Function prototype</b>	void spi_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable SPI interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<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:

```
/* disable SPI transmit buffer empty interrupt */
spi_interrupt_disable(SPI_INT_TBE);
```

## spi\_interrupt\_flag\_get

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

**Table 3-807. Function spi\_interrupt\_flag\_get**

<b>Function name</b>	spi_interrupt_flag_get
<b>Function prototype</b>	FlagStatus spi_interrupt_flag_get(uint8_t int_flag);
<b>Function descriptions</b>	get SPI interrupt status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```

/* get SPI transmit buffer empty interrupt status */
if(RESET != spi_interrupt_flag_get(SPI_INT_FLAG_TBE)){
    while(RESET == spi_flag_get(SPI_FLAG_TBE));
    spi_data_transmit(SPI_send_array[send_n++]);
}

```

## 3.25. SVPWM

The SVPWM registers are listed in chapter [3.26.1](#), and the SVPWM firmware functions are introduced in chapter [3.26.2](#).

### 3.25.1. Descriptions of Peripheral registers

SVPWM registers are listed in the table shown as below:



**Table 3-808. SVPWM Registers**

Registers	Descriptions
SVPWM_CTL0	control register 0
SVPWM_UALPHA	voltage alpha register
SVPWM_UBETA	voltage beta register
SVPWM_CAR	counter reload register
SVPWM_TA	ta register
SVPWM_TB	tb register
SVPWM_TC	tc register
SVPWM_SEC	sector register
SVPWM_STAT	status register

### 3.25.2. Descriptions of Peripheral functions

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

**Table 3-809. SVPWM firmware function**

Function name	Function description
svpwm_deinit	initialize the parameters of SVPWM struct
svpwm_init	initialize SVPWM
svpwm_enable	enable the SVPWM
svpwm_flag_get	check the SVPWM status flag
svpwm_alpha_beta_write	write alpha and beta data in iq format
svpwm_ta_tb_tc_read	read ta, tb, tc data in uint16 forma
svpwm_sector_read	read sector data

#### Structure svpwm\_parameter\_struct

**Table 3-810. Structure svpwm\_parameter\_struct**

Member name	Function description
switch_mode	SVPWM result_mode (SVPWM_SWITCH_MODE0, SVPWM_SWITCH_MODE1)
working_mode	SVPWM mode definitions (SVPWM_MODE_SEVEN_SEGMENT, SVPWM_MODE_FIVE_SEGMENT)
period_count	period_count

#### svpwm\_deinit

The description of svpwm\_deinit is shown as below:

**Table 3-811. Function svpwm\_deinit**

Function name	svpwm_deinit
Function prototype	void svpwm_deinit(void);
Function descriptions	reset the SVPWM
Precondition	-

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

Example:

```
/* reset SVPWM registers */
```

```
svpwm_deinit();
```

### svpwm\_init

The description of svpwm\_init is shown as below:

**Table 3-812. Function svpwm\_init**

Function name	svpwm_init
Function prototype	void svpwm_init(svpwm_parameter_struct *init_struct);
Function descriptions	initialize SVPWM
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	refer to <a href="#">Structure svpwm_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize SVPWM */
```

```
svpwm_parameter_struct init_struct;
```

```
svpwm_init (&init_struct);
```

### svpwm\_enable

The description of svpwm\_enable is shown as below:

**Table 3-813. Function svpwm\_enable**

Function name	svpwm_enable
Function prototype	void svpwm_enable(void);
Function descriptions	enable the SVPWM
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* enable the SVPWM */
```

```
svpwm_enable ();
```

### svpwm\_flag\_get

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

**Table 3-814. Function svpwm\_flag\_get**

<b>Function name</b>	svpwm_flag_get
<b>Function prototype</b>	FlagStatus svpwm_flag_get(uint32_t flag);
<b>Function descriptions</b>	check the SVPWM status flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>flag</b>	the timer interrupt flags
<i>SVPWM_FLAG_OAL</i>	the duty cycle of A-phase is less than 0 overflow flag
<i>SVPWM_FLAG_OBL</i>	the duty cycle of B-phase is less than 0 overflow flag
<i>SVPWM_FLAG_OCL</i>	the duty cycle of C-phase is less than 0 overflow flag
<i>SVPWM_FLAG_OAH</i>	the duty cycle of A-phase is greater than the COUNT overflow flag
<i>SVPWM_FLAG_OBH</i>	the duty cycle of B-phase is greater than the COUNT overflow flag
<i>SVPWM_FLAG_OCH</i>	the duty cycle of C-phase is greater than the COUNT overflow flag
<i>SVPWM_FLAG_OSF</i>	the svpwm operation is executed successfully flag
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get SVPWM status flag */
```

```
FlagStatus Flag_status = RESET;
```

```
Flag_status = svpwm_flag_get (SVPWM_FLAG_OAL);
```

### svpwm\_alpha\_beta\_write

The description of svpwm\_alpha\_beta\_write is shown as below:

Table 3-815. Function svpwm\_alpha\_beta\_write

Function name	svpwm_alpha_beta_write
Function prototype	void svpwm_alpha_beta_write(float alpha, float beta);
Function descriptions	write alpha and beta data in iq format
Precondition	-
The called functions	-
Input parameter{in}	
alpha	voltage alpha value
Input parameter{in}	
beta	voltage beta value
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write alpha and beta data in iq format */
```

```
svpwm_alpha_beta_write (1000,1000);
```

### svpwm\_ta\_tb\_tc\_read

The description of svpwm\_ta\_tb\_tc\_read is shown as below:

Table 3-816. Function svpwm\_ta\_tb\_tc\_read

Function name	svpwm_ta_tb_tc_read
Function prototype	void svpwm_ta_tb_tc_read(uint16_t *ta, uint16_t *tb, uint16_t *tc);
Function descriptions	read ta tb tc.
Precondition	-
The called functions	-
Input parameter{in}	
*ta	The address of ta
Input parameter{in}	
*tb	The address of tb
Input parameter{in}	
*tc	The address of tc
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read ta, tb, tc data in uint16 format */
```

```
uint16_t ta,tb,tc;
```

```
svpwm_ta_tb_tc_read (&ta,&tb,&tc);
```

## svpwm\_sector\_read

The description of `svpwm_sector_read` is shown as below:

**Table 3-817. Function `svpwm_sector_read`**

<b>Function name</b>	<code>svpwm_sector_read</code>
<b>Function prototype</b>	<code>uint8_t svpwm_sector_read();</code>
<b>Function descriptions</b>	read sector data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<code>uint8_t</code>	the sector output data

Example:

```
/* read sector data */
```

```
uint8_t sector;
```

```
sector = svpwm_sector_read();
```

## 3.26. SYSCFG

The SYSCFG registers are listed in chapter [3.26.1](#), and the SYSCFG firmware functions are introduced in chapter [3.26.2](#).

### 3.26.1. Descriptions of Peripheral registers

SYSCFG registers are listed in the table shown as below:

**Table 3-818. SYSCFG Registers**

Registers	Descriptions
SYSCFG_CFG0	configuration register 0
SYSCFG_CFG1	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_STAT	status register
SYSCFG_CFG2	configuration register 2

Registers	Descriptions
SYSCFG_CFG3	configuration register 3
SYSCFG_CFG4	configuration register 4
SYSCFG_TIMERCISEL0	TIMER input selection register 0
SYSCFG_TIMERCISEL1	TIMER input selection register 1
SYSCFG_CFG5	configuration register 5
SYSCFG_SRAMCFG	SRAM configuration register
SYSCFG_PRCFG	Protection configuration register
SYSCFG_TADSRCFG	TIMERx ADC start request configuration register
SYSCFG_TIMER0TRGCFG0	TIMERx configuration register0, x=0
SYSCFG_TIMER0TRGCFG1	TIMERx configuration register1, x=0
SYSCFG_TIMER0TRGCFG2	TIMERx configuration register2, x=0
SYSCFG_TIMER1TRGCFG0	TIMERx configuration register0, x=1
SYSCFG_TIMER1TRGCFG1	TIMERx configuration register1, x=1
SYSCFG_TIMER1TRGCFG2	TIMERx configuration register2, x=1
SYSCFG_TIMER2TRGCFG0	TIMERx configuration register0, x=2
SYSCFG_TIMER2TRGCFG1	TIMERx configuration register1, x=2
SYSCFG_TIMER2TRGCFG2	TIMERx configuration register2, x=2
SYSCFG_TIMER7TRGCFG0	TIMERx configuration register0, x=7
SYSCFG_TIMER7TRGCFG1	TIMERx configuration register1, x=7
SYSCFG_TIMER7TRGCFG2	TIMERx configuration register2, x=7
SYSCFG_CFG6	configuration register 6

### 3.26.2. Descriptions of Peripheral functions

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

**Table 3-819. SYSCFG firmware function**

Function name	Function description
syscfg_deinit	reset the SYSCFG registers
syscfg_i2c_fast_mode_plus_enable	enable I2C fast mode plus
syscfg_i2c_fast_mode_plus_disable	disable I2C fast mode plus
syscfg_exti_line_config	configure the GPIO pin as EXTI Line
syscfg_timer_input_source_select	select timer channel input source
syscfg_lockup_enable	enable module lockup function (function can be disabled by system reset)
syscfg_sram_remap_set	SRAM remap set
syscfg_sram_remap_size_get	get SRAM remap size
syscfg_ck_cmp_sel_set	set CK_CMP selection
syscfg_ck_cmp_sel_get	get CK_CMP selection
syscfg_register_write_enable	enable register write function
syscfg_register_write_disable	disable register write function
syscfg_adc_signal_monitor_select	ADC signal monitor select

Function name	Function description
syscfg_adc_signal_monitor_output_enable	ADC signal monitor output enable
syscfg_adc_signal_monitor_output_disable	ADC signal monitor output disable
syscfg_boot_mode_get	get boot mode
syscfg_sram_ecc_error_address_get	get sram ecc error address
syscfg_sram_ecc_error_bit_get	get sram ecc single-bit correction error bit
syscfg_interrupt_enable	enable the syscfg peripherals interrupt
syscfg_interrupt_disable	disable the syscfg peripherals interrupt
syscfg_interrupt_flag_get	get the interrupt flags
syscfg_interrupt_flag_clear	clear the interrupt flags

### Enum exti\_gpio\_enum

Table 3-820. Enum exti\_gpio\_enum

enum name	Function description
EXTI_SOURCE_EXTI0_PA8	EXTI0 PA8
EXTI_SOURCE_EXTI0_PC10	EXTI0 PC10
EXTI_SOURCE_EXTI0_PN2	EXTI0 PN2
EXTI_SOURCE_EXTI0_PC8	EXTI0 PC8
EXTI_SOURCE_EXTI1_PA9	EXTI1 PA9
EXTI_SOURCE_EXTI1_PC11	EXTI1 PC11
EXTI_SOURCE_EXTI1_PE13	EXTI1 PE13
EXTI_SOURCE_EXTI1_PC9	EXTI1 PC9
EXTI_SOURCE_EXTI2_PA0	EXTI2 PA0
EXTI_SOURCE_EXTI2_PC12	EXTI2 PC12
EXTI_SOURCE_EXTI2_PF14	EXTI2 PF14
EXTI_SOURCE_EXTI2_PG12	EXTI2 PG12
EXTI_SOURCE_EXTI3_PF12	EXTI3 PF12
EXTI_SOURCE_EXTI4_PA1	EXTI4 PA1
EXTI_SOURCE_EXTI4_PE14	EXTI4 PE14
EXTI_SOURCE_EXTI4_PF9	EXTI4 PF9
EXTI_SOURCE_EXTI4_PF13	EXTI4 PF13
EXTI_SOURCE_EXTI5_PD8	EXTI5 PD8
EXTI_SOURCE_EXTI5_PG14	EXTI5 PG14
EXTI_SOURCE_EXTI5_PN7	EXTI5 PN7
EXTI_SOURCE_EXTI6_PB1	EXTI6 PB1
EXTI_SOURCE_EXTI6_PC6	EXTI6 PC6
EXTI_SOURCE_EXTI6_PD2	EXTI6 PD2
EXTI_SOURCE_EXTI6_PG13	EXTI6 PG13
EXTI_SOURCE_EXTI7_PB0	EXTI7 PB0
EXTI_SOURCE_EXTI7_PC7	EXTI7 PC7
EXTI_SOURCE_EXTI7_PN5	EXTI7 PN5
EXTI_SOURCE_EXTI8_PD4	EXTI8 PD4

enum name	Function description
EXTI_SOURCE_EXTI8_PE8	EXTI8 PE8
EXTI_SOURCE_EXTI8_PF8	EXTI8 PF8
EXTI_SOURCE_EXTI8_PG15	EXTI8 PG15
EXTI_SOURCE_EXTI9_PD5	EXTI9 PD5
EXTI_SOURCE_EXTI9_PD9	EXTI9 PD9
EXTI_SOURCE_EXTI9_PE9	EXTI9 PE9
EXTI_SOURCE_EXTI9_PF11	EXTI9 PF11
EXTI_SOURCE_EXTI10_PB2	EXTI10 PB2
EXTI_SOURCE_EXTI10_PC0	EXTI10 PC0
EXTI_SOURCE_EXTI10_PD10	EXTI10 PD10
EXTI_SOURCE_EXTI10_PE10	EXTI10 PE10
EXTI_SOURCE_EXTI10_PF10	EXTI10 PF10
EXTI_SOURCE_EXTI11_PC1	EXTI11 PC1
EXTI_SOURCE_EXTI11_PD11	EXTI11 PD11
EXTI_SOURCE_EXTI11_PG11	EXTI11 PG11
EXTI_SOURCE_EXTI12_PC2	EXTI12 PC2
EXTI_SOURCE_EXTI12_PD12	EXTI12 PD12
EXTI_SOURCE_EXTI12_PE12	EXTI12 PE12
EXTI_SOURCE_EXTI13_PC3	EXTI13 PC3
EXTI_SOURCE_EXTI13_PD13	EXTI13 PD13
EXTI_SOURCE_EXTI14_PB14	EXTI14 PB14
EXTI_SOURCE_EXTI14_PC4	EXTI14 PC4
EXTI_SOURCE_EXTI14_PD14	EXTI14 PD14
EXTI_SOURCE_EXTI14_PE11	EXTI14 PE11
EXTI_SOURCE_EXTI15_PB15	EXTI15 PB15
EXTI_SOURCE_EXTI15_PC5	EXTI15 PC5

## Enum syscfg\_interrupt\_enum

Table 3-821. Enum syscfg\_interrupt\_enum

enum name	Function description
SYSCFG_INT_ECC_ME0	SRAM multi-bit non-correction interrupt enable
SYSCFG_INT_ECC_SE0	SRAM single-bit correction interrupt enable
SYSCFG_INT_ECC_ME1	CAN receive FIFO multi-bit non-correction interrupt enable
SYSCFG_INT_ECC_SE1	CAN receive FIFO single-bit correction interrupt enable
SYSCFG_INT_ECC_ME2	CAN filter SRAM multi-bit non-correction interrupt enable
SYSCFG_INT_ECC_SE2	CAN filter SRAM single-bit correction interrupt enable
SYSCFG_INT_ECC_ME3	Flash SRAM multi-bit non-correction interrupt enable
SYSCFG_INT_ECC_SE3	Flash SRAM single-bit correction interrupt enable
SYSCFG_INT_ECC_ME4	Flash multi-bit non-correction interrupt enable
SYSCFG_INT_NMI_HXTAL	HXTAL clock monitor NMI interrupt enable
SYSCFG_INT_NMI_PIN	NMI pin interrupt enable



enum name	Function description
SYSCFG_INT_NMI_WWDG T	WWDGT NMI interrupt enable
SYSCFG_INT_NMI_FWDGT	FWDGT NMI interrupt enable
SYSCFG_INT_NMI_LVD2	LVD2 NMI interrupt enable
SYSCFG_INT_NMI_LVD1	LVD1 NMI interrupt enable

### Enum syscfg\_adcsn\_enum

**Table 3-822. Enum syscfg\_adcsn\_enum**

enum name	Function description
SYSCFG_ADCSM1_TIMER0_TRGO F	ADCSM1 Source of monitor is TIMER0_TRGOF
SYSCFG_ADCSM1_TIMER0_TRGU F	ADCSM1 Source of monitor is TIMER0_TRGUF
SYSCFG_ADCSM1_TIMER0_TRGA	ADCSM1 Source of monitor is TIMER0_TRGA
SYSCFG_ADCSM1_TIMER0_TRGB	ADCSM1 Source of monitor is TIMER0_TRGB
SYSCFG_ADCSM1_TIMER0_TRGA B	ADCSM1 Source of monitor is TIMER0_TRGAB
SYSCFG_ADCSM1_TIMER7_TRGO F	ADCSM1 Source of monitor is TIMER7_TRGOF
SYSCFG_ADCSM1_TIMER7_TRGU F	ADCSM1 Source of monitor is TIMER7_TRGUF
SYSCFG_ADCSM1_TIMER7_TRGA	ADCSM1 Source of monitor is TIMER7_TRGA
SYSCFG_ADCSM1_TIMER7_TRGB	ADCSM1 Source of monitor is TIMER7_TRGB
SYSCFG_ADCSM1_TIMER7_TRGA B	ADCSM1 Source of monitor is TIMER7_TRGAB
SYSCFG_ADCSM2_TIMER0_TRGO F	ADCSM2 Source of monitor is TIMER0_TRGOF
SYSCFG_ADCSM2_TIMER0_TRGU F	ADCSM2 Source of monitor is TIMER0_TRGUF
SYSCFG_ADCSM2_TIMER0_TRGA	ADCSM2 Source of monitor is TIMER0_TRGA
SYSCFG_ADCSM2_TIMER0_TRGB	ADCSM2 Source of monitor is TIMER0_TRGB
SYSCFG_ADCSM2_TIMER0_TRGA B	ADCSM2 Source of monitor is TIMER0_TRGAB
SYSCFG_ADCSM2_TIMER7_TRGO F	ADCSM2 Source of monitor is TIMER7_TRGOF
SYSCFG_ADCSM2_TIMER7_TRGU F	ADCSM2 Source of monitor is TIMER7_TRGUF
SYSCFG_ADCSM2_TIMER7_TRGA	ADCSM2 Source of monitor is TIMER7_TRGA
SYSCFG_ADCSM2_TIMER7_TRGB	ADCSM2 Source of monitor is TIMER7_TRGB
SYSCFG_ADCSM2_TIMER7_TRGA B	ADCSM2 Source of monitor is TIMER7_TRGAB

## Enum timer\_channel\_input\_enum

**Table 3-823. Enum timer\_channel\_input\_enum**

enum name	Function description
TIMER7_Ci0_INPUT_TIMER7_CH0	select TIMER7 CH0 as TIMER7 Ci0
TIMER7_Ci0_INPUT_CMP1_OUT	select CMP1 output as TIMER7 Ci0
TIMER7_Ci1_INPUT_TIMER7_CH1	select TIMER7 CH1 as TIMER7 Ci1
TIMER7_Ci2_INPUT_TIMER7_CH2	select TIMER7 CH2 as TIMER7 Ci2
TIMER7_Ci3_INPUT_TIMER7_CH3	select TIMER7 CH3 as TIMER7 Ci3
TIMER0_Ci0_INPUT_TIMER0_CH0	select TIMER0 CH0 as TIMER0 Ci0
TIMER0_Ci0_INPUT_CMP0_OUT	select CMP0 output as TIMER0 Ci0
TIMER0_Ci1_INPUT_TIMER0_CH1	select TIMER0 CH1 as TIMER0 Ci1
TIMER0_Ci2_INPUT_TIMER0_CH2	select TIMER0 CH2 as TIMER0 Ci2
TIMER0_Ci3_INPUT_TIMER0_CH3	select TIMER0 CH3 as TIMER0 Ci3
TIMER2_Ci0_INPUT_TIMER2_CH0	select TIMER2 CH0 as TIMER2 Ci0
TIMER2_Ci0_INPUT_CMP0_OUT	select CMP0 output as TIMER2 Ci0
TIMER2_Ci0_INPUT_CMP1_OUT	select CMP1 output as TIMER2 Ci0
TIMER2_Ci0_INPUT_CMP0_1_OUT	select CMP0 and CMP1 output as TIMER2 Ci0
TIMER2_Ci0_INPUT_EXT_TRIGGER0	select external trigger 0 as TIMER2 Ci0
TIMER2_Ci0_INPUT_EXT_TRIGGER1	select external trigger 1 as TIMER2 Ci0
TIMER2_Ci0_INPUT_EXT_TRIGGER2	select external trigger 2 as TIMER2 Ci0
TIMER2_Ci0_INPUT_EXT_TRIGGER3	select external trigger 3 as TIMER2 Ci0
TIMER2_Ci1_INPUT_TIMER2_CH1	select TIMER2 CH1 as TIMER2 Ci1
TIMER2_Ci1_INPUT_EXT_TRIGGER0	select external trigger 0 as TIMER2 Ci1
TIMER2_Ci1_INPUT_EXT_TRIGGER1	select external trigger 1 as TIMER2 Ci1
TIMER2_Ci1_INPUT_EXT_TRIGGER2	select external trigger 2 as TIMER2 Ci1
TIMER2_Ci1_INPUT_EXT_TRIGGER3	select external trigger 3 as TIMER2 Ci1
TIMER2_Ci2_INPUT_TIMER2_CH2	select TIMER2 CH2 as TIMER2 Ci2
TIMER2_Ci3_INPUT_TIMER2_CH3	select TIMER2 CH3 as TIMER2 Ci3
TIMER2_Ci3_INPUT_IRC32K	select IRC32K as TIMER2 Ci3
TIMER2_Ci3_INPUT_CK_OUT	select Clock out as TIMER2 Ci3
TIMER1_Ci0_INPUT_TIMER1_CH0	select TIMER1 CH0 as TIMER1 Ci0
TIMER1_Ci0_INPUT_EXT_TRIGGER0	select external trigger 0 as TIMER1 Ci0
TIMER1_Ci0_INPUT_EXT_TRIGGER1	select external trigger 1 as TIMER1 Ci0
TIMER1_Ci0_INPUT_EXT_TRIGGER2	select external trigger 2 as TIMER1 Ci0
TIMER1_Ci0_INPUT_EXT_TRIGGER3	select external trigger 3 as TIMER1 Ci0
TIMER1_Ci1_INPUT_TIMER1_CH1	select TIMER1 CH1 as TIMER1 Ci1
TIMER1_Ci1_INPUT_EXT_TRIGGER0	select external trigger 0 as TIMER1 Ci1
TIMER1_Ci1_INPUT_EXT_TRIGGER1	select external trigger 1 as TIMER1 Ci1
TIMER1_Ci1_INPUT_EXT_TRIGGER2	select external trigger 2 as TIMER1 Ci1
TIMER1_Ci1_INPUT_EXT_TRIGGER3	select external trigger 3 as TIMER1 Ci1
TIMER1_Ci2_INPUT_TIMER1_CH2	select TIMER1 CH2 as TIMER1 Ci2
TIMER1_Ci3_INPUT_TIMER1_CH3	select TIMER1 CH3 as TIMER1 Ci3

enum name	Function description
TIMER1_CI3_INPUT_CMP0_OUT	select CMP0 output as TIMER1 CI3
TIMER1_CI3_INPUT_CMP1_OUT	select CMP1 output as TIMER1 CI3
TIMER1_CI3_INPUT_CMP0_1_OUT	select CMP0 and CMP1 output as TIMER1 CI3

### Enum ck\_cmp\_sel\_enum

Table 3-824. Enum ck\_cmp\_sel\_enum

enum name	Function description
CK_CMP_DIV2	CK_CMP=fPCLK/2
CK_CMP_DIV4	CK_CMP=fPCLK/4

### syscfg\_deinit

The description of syscfg\_deinit is shown as below:

Table 3-825. 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 3-826. Function syscfg\_i2c\_fast\_mode\_plus\_enable

Function name	syscfg_i2c_fast_mode_plus_enable
Function prototype	void syscfg_i2c_fast_mode_plus_enable(void);
Function descriptions	enable I2C fast mode plus
Precondition	-
The called functions	-
Input parameter{in}	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C fast mode plus function */
syscfg_i2c_fast_mode_plus_enable();
```

### syscfg\_i2c\_fast\_mode\_plus\_disable

The description of syscfg\_i2c\_fast\_mode\_plus\_disable is shown as below:

**Table 3-827. Function syscfg\_i2c\_fast\_mode\_plus\_disable**

Function name	syscfg_i2c_fast_mode_plus_disable
Function prototype	void syscfg_i2c_fast_mode_plus_disable(void);
Function descriptions	disable I2C fast mode plus
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C fast mode plus function */
syscfg_i2c_fast_mode_plus_disable();
```

### syscfg\_exti\_line\_config

The description of syscfg\_exti\_line\_config is shown as below:

**Table 3-828. Function syscfg\_exti\_line\_config**

Function name	syscfg_exti_line_config
Function prototype	void syscfg_exti_line_config(exti_gpio_enum exti_gpion);
Function descriptions	configure the GPIO pin as EXTI Line
Precondition	-
The called functions	-
Input parameter{in}	
exti_gpio	specify the GPIO port used in EXTI, refer to <a href="#">Table 3-820. Enum exti_gpio_enum</a>
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure the PA8 pin as EXTI0 Line */
```

```
syscfg_exti_line_config(EXTI_SOURCE_EXTI0_PA8);
```

### syscfg\_timer\_input\_source\_select

The description of syscfg\_timer\_input\_source\_select is shown as below:

**Table 3-829. 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>	-
<b>Input parameter{in}</b>	
timer_input	TIMER channel input select, refer to <a href="#">Table 3-821. Enum syscfg_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select timer channel input source */
```

```
syscfg_timer_input_source_select(TIMER7_CIO_INPUT_TIMER7_CH0);
```

### syscfg\_lockup\_enable

The description of syscfg\_lockup\_enable is shown as below:

**Table 3-830. Function syscfg\_lockup\_enable**

<b>Function name</b>	syscfg_lockup_enable
<b>Function prototype</b>	void syscfg_lockup_enable(uint32_t lockup);
<b>Function descriptions</b>	configure module lockup
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
lockup	lockup configuration
SYSCFG_LOCKUP_LOCKUP	CPU lockup (Hardfault) output enable

SYSCFG_LVD_LOCKUP	Low voltage detector lock
SYSCFG_SRAM_LOCKUP	SRAM ECC multi-bit error lock
SYSCFG_CANRX_LOCK	CAN receive FIFO ECC multi-bit error lock
SYSCFG_CANFILTER_LOCK	CAN filter ECC multi-bit error lock
SYSCFG_FLASHSRAM_LOCK	Flash SRAM ECC multi-bit error lock
SYSCFG_FLASH_LOCKUP	Flash ECC multi-bit error lock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable module lockup function */
```

```
syscfg_lockup_enable(SYSCFG_LOCKUP_LOCKUP);
```

### syscfg\_sram\_remap\_set

The description of syscfg\_sram\_remap\_set is shown as below:

**Table 3-831. syscfg\_sram\_remap\_set**

<b>Function name</b>	syscfg_sram_remap_set
<b>Function prototype</b>	void syscfg_sram_remap_set(uint32_t size);
<b>Function descriptions</b>	set SRAM remap size
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>size</b>	SRAM remap size
SRAM_REMAP_SIZE_0KB	0KB
SRAM_REMAP_SIZE_8KB	8KB
SRAM_REMAP_SIZE_16KB	16KB
SRAM_REMAP_SIZE_32KB	32KB
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set sram remap to 32KB */
```

```
syscfg_sram_remap_set(SRAM_REMAP_SIZE_32KB);
```

### syscfg\_sram\_remap\_size\_get

The description of syscfg\_sram\_remap\_size\_get is shown as below:

Table 3-832. Function syscfg\_sram\_remap\_size\_get

Function name	syscfg_sram_remap_size_get
Function prototype	uint32_t syscfg_sram_remap_size_get(void);
Function descriptions	get SRAM remap size
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	get SRAM remap size
SRAM_REMAP_SIZE_0KB	0KB
SRAM_REMAP_SIZE_8KB	8KB
SRAM_REMAP_SIZE_16KB	16KB
SRAM_REMAP_SIZE_32KB	32KB

Example:

```
uint32_t size;
```

```
size = syscfg_sram_remap_size_get();
```

### syscfg\_ck\_cmp\_sel\_set

The description of syscfg\_ck\_cmp\_sel\_set is shown as below:

Table 3-833. syscfg\_ck\_cmp\_sel\_set

Function name	syscfg_ck_cmp_sel_set
Function prototype	void syscfg_ck_cmp_sel_set(ck_cmp_sel_enum sel)
Function descriptions	set CK_CMP selection
Precondition	-
The called functions	-
Input parameter{in}	
sel	CK_CMP selection
CK_CMP_DIV2	CK_CMP=fPCLK/2
CK_CMP_DIV4	CK_CMP=fPCLK/4
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set CK_CMP=fPCLK/4 */
```

```
syscfg_ck_cmp_sel_set(CK_CMP_DIV4);
```

## syscfg\_ck\_cmp\_sel\_get

The description of syscfg\_ck\_cmp\_sel\_get is shown as below:

**Table 3-834. Function syscfg\_ck\_cmp\_sel\_get**

<b>Function name</b>	syscfg_ck_cmp_sel_get
<b>Function prototype</b>	ck_cmp_sel_enum syscfg_ck_cmp_sel_get(void)
<b>Function descriptions</b>	get CK_CMP selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>sel</b>	CK_CMP selection
CK_CMP_DIV2	CK_CMP=fPCLK/2
CK_CMP_DIV4	CK_CMP=fPCLK/4

Example:

```
ck_cmp_sel_enum sel;

sel = syscfg_ck_cmp_sel_get();
```

## syscfg\_register\_write\_enable

The description of syscfg\_register\_write\_enable is shown as below:

**Table 3-835. syscfg\_register\_write\_enable**

<b>Function name</b>	syscfg_register_write_enable
<b>Function prototype</b>	void syscfg_register_write_enable(uint32_t wp_reg);
<b>Function descriptions</b>	enable register write function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wp_reg</b>	write protected register
SYSCFG_WRITE_PROTECTION_REG0	RCU_CTL, RCU_CFG0, RCU_INT, RCU_CFG1, RCU_CFG2
SYSCFG_WRITE_PROTECTION_REG1	RCU_APB1RST, RCU_APB2RST, RCU_AHBEN, RCU_APB1EN, RCU_APB2EN, RCU_RSTSCK, RCU_AHBRST
SYSCFG_WRITE_PROTECTION_REG2	PMU_LVD1CTL, PMU_LVD2CTL
<b>Output parameter{out}</b>	



-	-
Return value	
-	-

Example:

```
/* enable write for RCU_CTL, RCU_CFG0, RCU_INT, RCU_CFG1, RCU_CFG2 */
syscfg_register_write_enable(SYSCFG_WRITE_PROTECTION_REG0);
```

### syscfg\_register\_write\_disable

The description of syscfg\_register\_write\_disable is shown as below:

**Table 3-836. syscfg\_register\_write\_disable**

<b>Function name</b>	syscfg_register_write_disable
<b>Function prototype</b>	void syscfg_register_write_disable(uint32_t wp_reg);
<b>Function descriptions</b>	enable register write protection function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wp_reg</b>	write protected register
SYSCFG_WRITE_PROTECTION_REG0	RCU_CTL, RCU_CFG0, RCU_INT, RCU_CFG1, RCU_CFG2
SYSCFG_WRITE_PROTECTION_REG1	RCU_APB1RST, RCU_APB2RST, RCU_AHBEN, RCU_APB1EN, RCU_APB2EN, RCU_RSTSCK, RCU_AHBRST
SYSCFG_WRITE_PROTECTION_REG2	PMU_LVD1CTL, PMU_LVD2CTL
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable write for RCU_CTL, RCU_CFG0, RCU_INT, RCU_CFG1, RCU_CFG2 */
syscfg_register_write_disable(SYSCFG_WRITE_PROTECTION_REG0);
```

### syscfg\_adc\_signal\_monitor\_select

The description of syscfg\_adc\_signal\_monitor\_select is shown as below:

**Table 3-837. syscfg\_adc\_signal\_monitor\_select**

<b>Function name</b>	syscfg_adc_signal_monitor_select
<b>Function prototype</b>	void syscfg_adc_signal_monitor_select(syscfg_adcsn_enum timer_signal);

<b>Function descriptions</b>	select ADC signal monitor
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_signal</b>	TIMER signal source, refer to <a href="#">Table 3-822. Enum syscfg_adcsm_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config and enable TIMER0_TRGOF output to ADCSM1 */
syscfg_adc_signal_monitor_select(SYSCFG_ADCSM1_TIMER0_TRGOF);
syscfg_adc_signal_monitor_output_enable(SYSCFG_TADSRCFG_ADCSM1);
```

### syscfg\_adc\_signal\_monitor\_output\_enable

The description of syscfg\_adc\_signal\_monitor\_output\_enable is shown as below:

**Table 3-838. syscfg\_adc\_signal\_monitor\_output\_enable**

<b>Function name</b>	syscfg_adc_signal_monitor_output_enable
<b>Function prototype</b>	void syscfg_adc_signal_monitor_output_enable(uint32_t adcsn);
<b>Function descriptions</b>	enable ADC signal monitor output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adcsn</b>	ADC signal monitor
SYSCFG_TADSRCFG_ADCSM1	ADC SM1 pin output
SYSCFG_TADSRCFG_ADCSM2	ADC SM2 pin output
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config and enable TIMER0_TRGOF output to ADCSM1 */
syscfg_adc_signal_monitor_select(SYSCFG_ADCSM1_TIMER0_TRGOF);
syscfg_adc_signal_monitor_output_enable(SYSCFG_TADSRCFG_ADCSM1);
```

### syscfg\_adc\_signal\_monitor\_output\_disable

The description of syscfg\_adc\_signal\_monitor\_output\_disable is shown as below:

Table 3-839. syscfg\_adc\_signal\_monitor\_output\_disable

Function name	syscfg_adc_signal_monitor_output_disable
Function prototype	void syscfg_adc_signal_monitor_output_disable(uint32_t adcsn);
Function descriptions	disable ADC signal monitor output
Precondition	-
The called functions	-
Input parameter{in}	
adcsn	ADC signal monitor
SYSCFG_TADSRCFG_ADCSM1	ADC SM1 pin output
SYSCFG_TADSRCFG_ADCSM2	ADC SM2 pin output
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable output to ADCSM1 */
```

```
syscfg_adc_signal_monitor_output_disable(SYSCFG_TADSRCFG_ADCSM1);
```

### syscfg\_boot\_mode\_get

The description of syscfg\_boot\_mode\_get is shown as below:

Table 3-840. syscfg\_boot\_mode\_get

Function name	syscfg_boot_mode_get
Function prototype	uint32_t syscfg_boot_mode_get(void);
Function descriptions	get boot mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	boot mode
SYSCFG_BOOT_SYSTEM_FLASH	Boot from the system memory
SYSCFG_BOOT_MAIN_FLASH	Boot from the main flash

Example:

```
/* get boot mode */
```

```
uint32_t boot_mode=0U;
```

```
boot_mode = syscfg_boot_mode_get();
```

### syscfg\_sram\_ecc\_error\_address\_get

The description of syscfg\_sram\_ecc\_error\_address\_get is shown as below:

**Table 3-841. 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(void);
<b>Function descriptions</b>	get SRAM ECC error address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>addr</b>	error address, 0-0x1FFF

Example:

```
/* get SRAM ECC error address */
```

```
uint32_t error_address = 0U;
```

```
error_address = syscfg_sram_ecc_error_address_get();
```

### syscfg\_sram\_ecc\_error\_bit\_get

The description of syscfg\_sram\_ecc\_error\_bit\_get is shown as below:

**Table 3-842. Function syscfg\_sram\_ecc\_error\_bit\_get**

<b>Function name</b>	syscfg_sram_ecc_error_bit_get
<b>Function prototype</b>	uint8_t syscfg_sram_ecc_error_bit_get(void);
<b>Function descriptions</b>	get sram ecc single-bit correction error bit
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>addr</b>	error address, 0-0x1FFF

Example:

```
/* get SRAM ECC error address */
```

```
uint32_t error_address = 0U;

error_address = syscfg_sram_ecc_error_address_get();
```

### syscfg\_interrupt\_enable

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

**Table 3-843. 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 3-821. Enum syscfg_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the SRAM ECC multi-bits non-correction event flag */

syscfg_interrupt_enable(SYSCFG_INT_ECC_ME0);
```

### syscfg\_interrupt\_disable

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

**Table 3-844. 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 3-821. Enum syscfg_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* Disable the SRAM ECC multi-bits non-correction event flag */

syscfg_interrupt_disable(SYSCFG_INT_ECC_ME0);
```

## syscfg\_interrupt\_flag\_get

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

**Table 3-845. Function syscfg\_interrupt\_flag\_get**

<b>Function name</b>	syscfg_interrupt_flag_get
<b>Function prototype</b>	FlagStatus syscfg_interrupt_flag_get(uint32_t 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
SYSCFG_INT_FLAG_ECC_ME0	SRAM multi-bit error flag
SYSCFG_INT_FLAG_ECC_SE0	SRAM single-bit error flag
SYSCFG_INT_FLAG_ECC_ME1	CAN FIFO SRAM multi-bit error flag
SYSCFG_INT_FLAG_ECC_SE1	CAN FIFO SRAM single-bit error flag
SYSCFG_INT_FLAG_ECC_ME2	CAN filter SRAM multi-bit error flag
SYSCFG_INT_FLAG_ECC_SE2	CAN filter SRAM single-bit error flag
SYSCFG_INT_FLAG_ECC_ME3	Flash SRAM multi-bit error flag
SYSCFG_INT_FLAG_ECC_SE3	Flash SRAM single-bit error flag
SYSCFG_INT_FLAG_ECC_ME4	Flash multi-bit error flag
SYSCFG_INT_FLAG_NMI_HXTAL	HXTAL clock monitor NMI interrupt flag
SYSCFG_INT_FLAG_NMI_PIN	NMI pin interrupt flag
SYSCFG_INT_FLAG_NMI_WWDGT	WWDGT NMI interrupt flag
SYSCFG_INT_FLAG_NMI_FWDGT	FWDGT NMI interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	Interrupt flag status
SET	Flag set
RESET	Flag is not set

Example:

```
/* get the SRAM ECC multi-bits non-correction event flag */
```

```
FlagStatus Flag = RESET;
```

```
Flag = syscfg_interrupt_flag_get(SYSCFG_INT_FLAG_ECC_ME0);
```

### syscfg\_interrupt\_flag\_clear

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

**Table 3-846. Function syscfg\_interrupt\_flag\_clear**

Function name	syscfg_interrupt_flag_clear
Function prototype	void syscfg_interrupt_flag_clear(uint32_t int_flag);
Function descriptions	clear interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
int_flag	Clear interrupt flag
SYSCFG_INT_FLAG_ECC_ME0	SRAM multi-bit error flag
SYSCFG_INT_FLAG_ECC_SE0	SRAM single-bit error flag
SYSCFG_INT_FLAG_ECC_ME1	CAN FIFO SRAM multi-bit error flag
SYSCFG_INT_FLAG_ECC_SE1	CAN FIFO SRAM single-bit error flag
SYSCFG_INT_FLAG_ECC_ME2	CAN filter SRAM multi-bit error flag
SYSCFG_INT_FLAG_ECC_SE2	CAN filter SRAM single-bit error flag
SYSCFG_INT_FLAG_ECC_ME3	Flash SRAM multi-bit error flag
SYSCFG_INT_FLAG_ECC_SE3	Flash SRAM single-bit error flag
SYSCFG_INT_FLAG_ECC_ME4	Flash multi-bit error flag
SYSCFG_INT_FLAG_NMI_FW DGT	FWDGT NMI interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the SRAM ECC multi-bits non-correction event flag */
```

```
syscfg_interrupt_flag_clear (SYSCFG_INT_FLAG_ECC_ME0);
```

## 3.27. 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 two sorts: advanced timer (TIMERx, x=0, 7), general level0 timer (TIMERx, x=1,2). The specific functions of different types of timer are different. The TIMER registers are listed in chapter [3.27.1](#), the TIMER firmware functions are introduced in chapter [3.27.2](#)

### 3.27.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

**Table 3-847. TIMERx Registers**

Registers	Descriptions
TIMERx_CTL0	Control register 0
TIMERx_CTL1	Control register 1
TIMERx_SMCFG	Slave mode configuration register
TIMERx_DMAINTE N	DMA and interrupt enable register
TIMERx_INTF	Interrupt flag register
TIMERx_SWEVG	Software event generation register
TIMERx_CHCTL0	Channel control register 0
TIMERx_CHCTL1	Channel control register 1
TIMERx_CHCTL2	Channel control register 2
TIMERx_CNT	Counter register
TIMERx_PSC	Prescaler register
TIMERx_CAR	Counter auto reload register
TIMERx_CREP	Counter repetition register 0
TIMERx_CH0CV	Channel 0 capture/compare value register
TIMERx_CH1CV	Channel 1 capture/compare value register
TIMERx_CH2CV	Channel 2 capture/compare value register
TIMERx_CH3CV	Channel 3 capture/compare value register
TIMERx_CCHP	Channel complementary protection register
TIMERx_MCHCTL0	TIMER multi mode channel control register 0
TIMERx_MCHCTL1	TIMER multi mode channel control register 1
TIMERx_MCHCTL2	TIMER multi mode channel control register 2
TIMERx_MCH0CV	TIMER multi mode channel 0 capture or compare value register
TIMERx_MCH1CV	TIMER multi mode channel 1 capture or compare value register
TIMERx_MCH2CV	TIMER multi mode channel 2 capture or compare value register
TIMERx_MCH3CV	TIMER multi mode channel 3 capture or compare value register
TIMERx_CH0COM V_ADD	TIMER channel 0 additional compare value register
TIMERx_CH1COM	TIMER channel 1 additional compare value register



Registers	Descriptions
V_ADD	
TIMERx_CH2COM V_ADD	TIMER channel 2 additional compare value register
TIMERx_CH3COM V_ADD	TIMER channel 3 additional compare value register
TIMERx_CTL2	TIMER control register 2
TIMERx_AFCTL	Alternate function control register
TIMERx_IREP	Interrupt Counter repetition register
TIMERx_ADCRCTL	ADC conversion request control register
TIMERx_ADCCR1	ADC trigger compare value register 1
TIMERx_ADCCR2	ADC trigger compare value register 2
TIMERx_ADCREP	ADC counter repetition register
TIMERx_ADCTL	ADC trigger control register
TIMERx_SHUPM0	Shadow update mode register 0
TIMERx_SHUPM1	Shadow update mode register 1
TIMERx_COUNSEL	Counter overflow or underflow mode selection register
TIMERx_DMACFG	DMA configuration register
TIMERx_DMATB	DMA transfer buffer register
TIMERx_CFG	Configuration register

### 3.27.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

**Table 3-848. TIMERx firmware function**

Function name	Function description
timer_deinit	deinit a TIMER
timer_struct_para_init	initialize TIMER init parameter struct with a default value
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_upif_backup_enable	enable upif backup
timer_upif_backup_disable	disable upif backup
timer_flow_interrupt_source_select	configure the timer flow interrupt source selection
timer_update_source_select	configure the timer update event source selection

Function name	Function description
timer_external_input_source_select	configure the TIMER ETI
timer_prescaler_config	configure TIMER prescaler
timer_update_repetition_value_config	configure timer update repetition register value
timer_flow_interrupt_repetition_value_config	configure timer flow interrupt repetition register value
timer_flow_interrupt_repetition_value_reload	timer flow interrupt repetition value immediate reload
timer_autoreload_value_config	configure TIMER autoreload register value
timer_autoreload_value_read	get TIMER autoreload register value
timer_counter_value_config	configure TIMER counter register value
timer_counter_read	read TIMER counter value
timer_prescaler_read	read TIMER prescaler value
timer_single_pulse_mode_config	configure TIMER single pulse mode
timer_update_source_config	configure TIMER update source
timer_channel_control_shadow_config	configure channel capture/compare control shadow register
timer_channel_control_shadow_update_config	configure TIMER channel control shadow register update control
timer_ocpre_clr_source_select	configure TIMER OCPRE_CLR source
timer_ocpre_clr_int_source_select	configure TIMER OCPRE_CLR_INT source selection
timer_channel_composite_asymmetric_pwm_level_config	configure TIMER channel period point match moment OxCPRE level in the composite PWM or asymmetric PWM mode
timer_dma_enable	enable the TIMER DMA
timer_dma_disable	disable the TIMER DMA
timer_channel_dma_request_source_select	select channel DMA request source
timer_dma_transfer_config	configure the TIMER DMA transfer
timer_event_software_generate	Configure software generate events
timer_break_struct_para_init	initialize TIMER break parameter struct with a default value
timer_break_config	configure TIMER break function
timer_break_enable	enable TIMER break function
timer_break_disable	disable TIMER break function
timer_automatic_output_enable	enable TIMER output automatic function
timer_automatic_output_disable	disable TIMER output automatic function
timer_primary_output_config	configure TIMER primary output function
timer_channel_output_struct_para_init	initialize TIMER channel output parameter struct with a default value
timer_channel_output_config	configure TIMER channel output function
timer_channel_output_mode_config	configure TIMER channel output compare mode
timer_channel_output_pulse_value_config	configure TIMER channel output pulse value

Function name	Function description
timer_channel_output_shadow_config	configure TIMER channel output shadow function
timer_channel_output_compare_fast_config	configure TIMER channel output compare fast function
timer_channel_output_clear_config	configure TIMER channel output clear function
timer_channel_output_polarity_config	configure TIMER channel output polarity
timer_channel_complementary_output_polarity_config	configure TIMER channel complementary output polarity
timer_channel_output_state_config	configure TIMER channel enable state
timer_channel_complementary_output_state_config	configure TIMER channel complementary output enable state
timer_channel_input_struct_parameter_init	initialize TIMER channel input parameter struct with a default value
timer_input_capture_config	configure TIMER input capture parameter
timer_channel_input_capture_prescaler_config	configure TIMER channel input capture prescaler value
timer_channel_capture_value_register_read	read TIMER channel capture compare register value
timer_input_pwm_capture_config	configure TIMER input pwm capture function
timer_hall_mode_config	configure TIMER hall sensor mode
timer_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	select multi mode channel mode
timer_input_trigger_source_select	select TIMER input trigger source
timer_master_output_trigger_source_select	select TIMER master mode output trigger source
timer_slave_mode_select	select TIMER slave mode
timer_master_slave_mode_config	configure TIMER master slave mode
timer_external_trigger_config	configure TIMER external trigger input
timer_quadrature_decoder_mode_config	configure TIMER quadrature decoder mode
timer_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_slave_mode_input_config	configure TIMER slave mode input

Function name	Function description
timer_external_clock_mode0_config	configure TIMER the external clock mode0
timer_external_clock_mode1_config	configure TIMER the external clock mode1
timer_external_clock_mode1_disable	disable TIMER the external clock mode1
timer_write_chxval_register_config	configure TIMER write CHxVAL register selection
timer_output_value_selection_config	configure TIMER output value selection
timer_output_match_pulse_select	configure TIMER output match pulse selection
timer_channel_composite_pwm_pulse_config	configure the TIMER composite PWM mode output pulse value
timer_channel_asymmetric_pwm_pulse_config	configure the TIMER asymmetric PWM mode output pulse value
timer_channel_additional_compare_value_config	configure TIMER channel additional compare value
timer_channel_additional_compare_value_read	read TIMER channel additional compare value
timer_channel_additional_output_shadow_config	configure TIMER channel additional output shadow function
timer_break_external_input_enable	enable break external input
timer_break_cmp_enable	enable break cmp
timer_break_external_input_disable	disable break external input
timer_break_cmp_disable	disable break cmp
timer_break_external_input_polarity_config	configure TIMER break external input polarity
timer_break_cmp_polarity_config	configure TIMER break cmp polarity
timer_break_auto_recover_event_select	Select break auto recover event
timer_trigger_adc_compare_enable	enable timer compare event produce a ADC converter trigger signal
timer_trigger_adc_compare_disable	disable timer compare event produce a ADC converter trigger signal
timer_trigger_adc_flow_enable	enable timer flow event produce a ADC converter trigger signal
timer_trigger_adc_flow_disable	disable timer flow event produce a ADC converter trigger signal
timer_trigger_adc_compare_value_config	configure adc trigger compare value register value
timer_trigger_adc_repetition_value_config	configure adc rep register value
timer_trigger_adc_repetition_decrement_select	select adc repetition_decrement source
timer_trigger_adc_repetition_value_reload	reload adc rep register value

Function name	Function description
reload	
timer_trigger_adc_compare_value_shadow_enable	enable adc compare register shadow function
timer_trigger_adc_compare_value_shadow_disable	disable adc compare register shadow function
timer_trigger_adc_monitor_config	configure ADC trigger signal monitoring function
timer_register_update_event_select	register reload event selection
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 flag
timer_interrupt_flag_clear	clear timer interrupt flag

### Structure timer\_parameter\_struct

**Table 3-849. 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
clockdivision	clock division value (TIMER_CKDIV_DIV1, TIMER_CKDIV_DIV2, TIMER_CKDIV_DIV4)
repetitioncounter	the counter repetition value (0~0xFF, use TIMER_CREP register)

### Structure timer\_break\_parameter\_struct

**Table 3-850. Structure timer\_break\_parameter\_struct**

Member name	Function description
runoffstate	run mode off-state(TIMER_ROS_STATE_ENABLE, TIMER_ROS_STATE_DISABLE)
ideloffstate	idle mode off-state(TIMER_IOS_STATE_ENABLE, TIMER_IOS_STATE_DISABLE)
deadtime	dead time(0~255)
breakpolarity	output automatic enable (TIMER_OUTAUTO_ENABLE, TIMER_OUTAUTO_DISABLE)
outputautostate	complementary register protect control(TIMER_CCHP0_PROT_OFF, TIMER_CCHP0_PROT_0, TIMER_CCHP0_PROT_1, TIMER_CCHP0_PROT_2)
protectmode	run mode off-state(TIMER_ROS_STATE_ENABLE,

	TIMER_ROS_STATE_DISABLE)
breakstate	BREAK input enable (TIMER_BREAK_ENABLE, TIMER_BREAK_DISABLE)
breakfilter	BREAK input filter(0~15)

### Structure timer\_oc\_parameter\_struct

**Table 3-851. 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 3-852. 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 3-853. 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)
----------	--

## timer\_deinit

The description of timer\_deinit is shown as below:

**Table 3-854. Function timer\_deinit**

<b>Function name</b>	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(x=0,1,2,7)</i>	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 3-855. 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 3-856. 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(x=0,1,2,7)</i>	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 3-857. Function timer\_enable

Function name	timer_enable
Function prototype	void timer_enable(uint32_t timer_periph);
Function descriptions	enable a TIMER
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,1,2,7)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 */
timer_enable(TIMER0);
```

### timer\_disable

The description of timer\_disable is shown as below:

Table 3-858. Function timer\_disable

Function name	timer_disable
Function prototype	void timer_disable(uint32_t timer_periph);
Function descriptions	disable a TIMER
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,1,2,7)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 */
timer_disable(TIMER0);
```

### timer\_auto\_reload\_shadow\_enable

The description of timer\_auto\_reload\_shadow\_enable is shown as below:

Table 3-859. 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(x=0,1,2,7)</i>	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 3-860. 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(x=0,1,2,7)</i>	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 3-861. 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
<i>TIMERx(x=0,1,2,7)</i>	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 3-862. 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
<i>TIMERx(x=0,1,2,7)</i>	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 3-863. 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=0,1,2,7)</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 3-864. 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>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

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 3-865. 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>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set TIMER0 counter down direction */
```

```
timer_counter_down_direction(TIMER0);
```

### timer\_upif\_backup\_enable

The description of timer\_upif\_backup\_enable is shown as below:

**Table 3-866. Function timer\_upif\_backup\_enable**

<b>Function name</b>	timer_upif_backup_enable
<b>Function prototype</b>	void timer_upif_backup_enable (uint32_t timer_periph);
<b>Function descriptions</b>	enable upif backup
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* enable upif backup */
timer_upif_backup_enable(TIMERO);
```

### timer\_upif\_backup\_disable

The description of timer\_upif\_backup\_disable is shown as below:

**Table 3-867. Function timer\_upif\_backup\_disable**

<b>Function name</b>	timer_upif_backup_disable
<b>Function prototype</b>	void timer_upif_backup_disable (uint32_t timer_periph);
<b>Function descriptions</b>	disable upif backup
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable upif backup */
timer_upif_backup_disable(TIMERO);
```

### timer\_flow\_interrupt\_source\_select

The description of timer\_flow\_interrupt\_source\_select is shown as below:

**Table 3-868. Function timer\_flow\_interrupt\_source\_select**

<b>Function name</b>	timer_flow_interrupt_source_select
<b>Function prototype</b>	void timer_flow_interrupt_source_select (uint32_t timer_periph, uint32_t source);
<b>Function descriptions</b>	configure the timer update event source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>source</b>	interrupt source

<i>TIMER_INT_SEL_FLOW</i>	Count overflow/underflow interrupt
<i>TIMER_INT_SEL_OVERFLOW</i>	Count overflow interrupt
<i>TIMER_INT_SEL_UNDERFLOW</i>	Count underflow interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the timer flow interrupt source selection */
```

```
timer_flow_interrupt_source_select(TIMER0, TIMER_INT_SEL_FLOW);
```

### timer\_update\_source\_select

The description of timer\_update\_source\_select is shown as below:

**Table 3-869. Function timer\_update\_source\_select**

<b>Function name</b>	timer_update_source_select
<b>Function prototype</b>	void timer_update_source_select (uint32_t timer_periph, uint32_t source);
<b>Function descriptions</b>	configure the timer update event source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>source</b>	update event source
<i>TIMER_UPS_SEL_FLOW</i>	Count overflow/underflow update
<i>TIMER_UPS_SEL_OVERFLOW</i>	Count overflow update
<i>TIMER_UPS_SEL_UNDERFLOW</i>	Count underflow update
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the timer update event source selection */
```

```
timer_update_source_select(TIMER0, TIMER_UPS_SEL_FLOW);
```

### timer\_external\_input\_source\_select

The description of timer\_external\_input\_source\_select is shown as below:

**Table 3-870. Function timer\_external\_input\_source\_select**

<b>Function name</b>	timer_external_input_source_select
<b>Function prototype</b>	void timer_external_input_source_select (uint32_t timer_periph, uint32_t source);
<b>Function descriptions</b>	configure the TIMER ETI
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>source</b>	external trigger input signal source
<i>TIMER_ETI_SEL_ETI1</i>	TIMER_ETI1, TIMERx(x = 0,1,2,7)
<i>TIMER_ETI_SEL_ETI2</i>	TIMER_ETI2, TIMERx(x = 0,1,2,7)
<i>TIMER_ETI_SEL_ETI3</i>	TIMER_ETI3, TIMERx(x = 1,2)
<i>TIMER_ETI_SEL_ETI4</i>	TIMER_ETI4, TIMERx(x = 1,2)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER ETI */
```

```
timer_external_input_source_select(TIMER0, TIMER_ETI_SEL_ETI1);
```

### timer\_prescaler\_config

The description of timer\_prescaler\_config is shown as below:

**Table 3-871. 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(x=0,7)</i>	TIMER peripheral selection



Input parameter{in}	
<b>prescaler</b>	prescaler value (0~0xFFFF)
Input parameter{in}	
<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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 prescaler */
```

```
timer_prescaler_config(TIMER0, 3000, TIMER_PSC_RELOAD_NOW);
```

### timer\_update\_repetition\_value\_config

The description of timer\_update\_repetition\_value\_config is shown as below:

**Table 3-872. Function timer\_update\_repetition\_value\_config**

<b>Function name</b>	timer_update_repetition_value_config
<b>Function prototype</b>	void timer_update_repetition_value_config (uint32_t timer_periph, uint16_t repetition)
<b>Function descriptions</b>	configure TIMER update repetition register value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
Input parameter{in}	
<b>repetition</b>	the counter repetition value (0~0xFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 update repetition register value */
```

```
timer_update_repetition_value_config(TIMER0, 98);
```

## timer\_flow\_interrupt\_repetition\_value\_config

The description of timer\_flow\_interrupt\_repetition\_value\_config is shown as below:

**Table 3-873. Function timer\_flow\_interrupt\_repetition\_value\_config**

<b>Function name</b>	timer_flow_interrupt_repetition_value_config
<b>Function prototype</b>	void timer_flow_interrupt_repetition_value_config(uint32_t timer_periph, uint16_t repetition)
<b>Function descriptions</b>	configure TIMER interrupt 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)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>repetition</b>	the counter repetition value (0~0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 interrupt repetition register value */
timer_flow_interrupt_repetition_value_config(TIMER0, 98);
```

## timer\_flow\_interrupt\_repetition\_value\_reload

The description of timer\_flow\_interrupt\_repetition\_value\_reload is shown as below:

**Table 3-874. Function timer\_flow\_interrupt\_repetition\_value\_reload**

<b>Function name</b>	timer_flow_interrupt_repetition_value_reload
<b>Function prototype</b>	void timer_flow_interrupt_repetition_value_reload (uint32_t timer_periph)
<b>Function descriptions</b>	reload timer flow interrupt repetition value immediate
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* timer flow interrupt repetition value immediate reload */
```

```
timer_flow_interrupt_repetition_value_reload(TIMERO);
```

### timer\_autoreload\_value\_config

The description of timer\_autoreload\_value\_config is shown as below:

**Table 3-875. 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, uint32_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=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>autoreload</b>	the integer part of auto-reload value, 0~0xFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER autoreload register value */
timer_autoreload_value_config(TIMERO, 3000);
```

### timer\_autoreload\_value\_read

The description of timer\_autoreload\_value\_read is shown as below:

**Table 3-876. 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=0,1,2,7)</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
--	----------

Example:

```
/* get TIMER autoreload register value */

uint32_t i = 0;

i =(uint32_t) timer_autoreload_value_read (TIMER0);
```

### timer\_counter\_value\_config

The description of timer\_counter\_value\_config is shown as below:

**Table 3-877. 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=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>counter</b>	the counter value 0~0xFFFF
<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 3-878. 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</i> ( <i>x</i> =0,1,2,7)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	counter value 0~0xFFFF

Example:

```
/* read TIMER0 counter value */

uint32_t i = 0;

i = timer_counter_read(TIMER0);
```

### timer\_prescaler\_read

The description of timer\_prescaler\_read is shown as below:

**Table 3-879. Function timer\_prescaler\_read**

<b>Function name</b>	timer_prescaler_read
<b>Function prototype</b>	uint16_t timer_prescaler_read(uint32_t timer_periph);
<b>Function descriptions</b>	read TIMER prescaler value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
timer_periph	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,1,2,7)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
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 3-880. Function timer\_single\_pulse\_mode\_config**

<b>Function name</b>	timer_single_pulse_mode_config
<b>Function prototype</b>	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=0,1,2,7)</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\_update\_source\_config

The description of timer\_update\_source\_config is shown as below:

**Table 3-881. 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=0,1,2,7)</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 – The counter generates an overflow or underflow event – The slave mode controller generates an update event
<i>TIMER_UPDATE_SRC_REGULAR</i>	only counter overflow/underflow generates an update interrupt or DMA request.
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* configure TIMER update only by counter overflow/underflow */
```

```
timer_update_source_config(TIMER0, TIMER_UPDATE_SRC_REGULAR);
```

### timer\_channel\_control\_shadow\_config

The description of timer\_channel\_control\_shadow\_config is shown as below:

**Table 3-882. Function timer\_channel\_control\_shadow\_config**

Function name	timer_channel_control_shadow_config
Function prototype	void timer_channel_control_shadow_config(uint32_t timer_periph, ControlStatus newvalue);
Function descriptions	configure channel commutation control shadow register
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* channel commutation control shadow register enable */
```

```
timer_channel_control_shadow_config(TIMER0, ENABLE);
```

### timer\_ocpre\_clr\_source\_select

The description of timer\_ocpre\_clr\_source\_select is shown as below:

**Table 3-883. Function timer\_ocpre\_clr\_source\_select**

Function name	timer_ocpre_clr_source_select
Function prototype	void timer_ocpre_clr_source_select(uint32_t timer_periph, uint32_t ocrsel);
Function descriptions	configure TIMER OCPRE_CLR source
Precondition	-
The called functions	-

Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
Input parameter{in}	
<b>ocrsel</b>	OCPRE_CLR source
<i>TIMER_OCPRE_CLR_CMP0</i>	OCPRE_CLR is connected to the CMP0
<i>TIMER_OCPRE_CLR_CMP1</i>	OCPRE_CLR is connected to the CMP1
<i>TIMER_OCPRE_CLR_CMP2</i>	OCPRE_CLR is connected to the CMP2
<i>TIMER_OCPRE_CLR_CMP3</i>	OCPRE_CLR is connected to the CMP3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TIMER0 OCPRE_CLR source is CMP0 */
```

```
timer_ocpre_clr_source_select(TIMER0, TIMER_OCPRE_CLR_CMP0);
```

### timer\_ocpre\_clr\_int\_source\_select

The description of timer\_ocpre\_clr\_int\_source\_select is shown as below:

**Table 3-884. Function timer\_ocpre\_clr\_int\_source\_select**

<b>Function name</b>	timer_ocpre_clr_int_source_select
<b>Function prototype</b>	void timer_ocpre_clr_int_source_select(uint32_t timer_periph, uint32_t selection);
<b>Function descriptions</b>	configure TIMER OCPRE_CLR_INT source
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
Input parameter{in}	
<b>selection</b>	OCPRE_CLR_INT source
<i>TIMER_SMCFG_OCPR_E_CLR_INPUT</i>	OCPRE_CLR_INT is connected to the OCPRE_CLR
<i>TIMER_SMCFG_OCPR_E_ETI</i>	OCPRE_CLR_INT is connected to the ETIF
Output parameter{out}	
-	-



Return value	
-	-

Example:

```
/* configure the TIMER0 OCPRE_CLR_INT is connected to the OCPRE_CLR input */
```

```
timer_ocpre_clr_int_source_select(TIMER0, TIMER_SMCFG_OCPRE_CLR_INPUT);
```

### timer\_channel\_composite\_asymmetric\_pwm\_level\_config

The description of timer\_channel\_composite\_asymmetric\_pwm\_level\_config is shown as below:

**Table 3-885. Function timer\_channel\_composite\_asymmetric\_pwm\_level\_config**

<b>Function name</b>	timer_channel_composite_asymmetric_pwm_level_config
<b>Function prototype</b>	void timer_channel_composite_asymmetric_pwm_level_config(uint32_t timer_periph, uint16_t channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure TIMER channel period point match moment OxCPRE level in the composite PWM or asymmetric PWM mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
<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 CH0 period point match moment OxCPRE level in the composite PWM or asymmetric PWM mode */
```

```
timer_channel_composite_asymmetric_pwm_level_config (TIMER0, TIMER_CH_0, ENABLE);
```

## timer\_dma\_enable

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

**Table 3-886. 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=0,1,2,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>dma</b>	specify which DMA to enable
<i>TIMER_DMA_UPD</i>	update DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CH0D</i>	channel 0 DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CH1D</i>	channel 1 DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CH2D</i>	channel 2 DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CH3D</i>	channel 3 DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CMTD</i>	commutation DMA request enable,TIMERx(x=0,7)
<i>TIMER_DMA_TRGD</i>	trigger DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_MCH0D</i>	multi mode channel 0 DMA request,TIMERx(x=0,7)
<i>TIMER_DMA_MCH1D</i>	multi mode channel 1 DMA request,TIMERx(x=0,7)
<i>TIMER_DMA_MCH2D</i>	multi mode channel 2 DMA request,TIMERx(x=0,7)
<i>TIMER_DMA_MCH3D</i>	multi mode channel 3 DMA request,TIMERx(x=0,7)
<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 3-887. 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(x=0,1,2,7)</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 enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CH0D</i>	channel 0 DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CH1D</i>	channel 1 DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CH2D</i>	channel 2 DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CH3D</i>	channel 3 DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_CMTD</i>	commutation DMA request enable,TIMERx(x=0,7)
<i>TIMER_DMA_TRGD</i>	trigger DMA enable,TIMERx(x=0,1,2,7)
<i>TIMER_DMA_MCH0D</i>	multi mode channel 0 DMA request,TIMERx(x=0,7)
<i>TIMER_DMA_MCH1D</i>	multi mode channel 1 DMA request,TIMERx(x=0,7)
<i>TIMER_DMA_MCH2D</i>	multi mode channel 2 DMA request,TIMERx(x=0,7)
<i>TIMER_DMA_MCH3D</i>	multi mode channel 3 DMA request,TIMERx(x=0,7)
<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 3-888. 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(x=0,1,2,7)</i>	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_CHANNEL_EVENT);
```

### timer\_dma\_transfer\_config

The description of timer\_dma\_transfer\_config is shown as below:

**Table 3-889. 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(x=0,1,2,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>dma_baseaddr</b>	DMA transfer access start address
<i>TIMER_DMACFG_DMA_TA_CTL0</i>	DMA transfer address is TIMER_CTL0, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA_TA_CTL1</i>	DMA transfer address is TIMER_CTL1, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA_TA_SMCFG</i>	DMA transfer address is TIMER_SMCFG, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA_TA_DMAINTEN</i>	DMA transfer address is TIMER_DMAINTEN, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA_TA_INTF</i>	DMA transfer address is TIMER_INTF, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA_TA_SWEVG</i>	DMA transfer address is TIMER_SWEVG, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA_TA_CHCTL0</i>	DMA transfer address is TIMER_CHCTL0, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA_TA_CHCTL1</i>	DMA transfer address is TIMER_CHCTL1, TIMERx(x=0,1,2,7)

<i>TIMER_DMACFG_DMA TA_CHCTL2</i>	DMA transfer address is TIMER_CHCTL2, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA TA_CNT</i>	DMA transfer address is TIMER_CNT, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA TA_PSC</i>	DMA transfer address is TIMER_PSC, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA TA_CAR</i>	DMA transfer address is TIMER_CAR, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA TA_CREP</i>	DMA transfer address is TIMER_CREP, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_CH0CV</i>	DMA transfer address is TIMER_CH0CV, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA TA_CH1CV</i>	DMA transfer address is TIMER_CH1CV, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA TA_CH2CV</i>	DMA transfer address is TIMER_CH2CV, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA TA_CH3CV</i>	DMA transfer address is TIMER_CH3CV, TIMERx(x=0,1,2,7)
<i>TIMER_DMACFG_DMA TA_CCHP</i>	DMA transfer address is TIMER_CCHP, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_MCHCTL0</i>	DMA transfer address is TIMER_MCHCTL0, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_MCHCTL1</i>	DMA transfer address is TIMER_MCHCTL1, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_MCHCTL2</i>	DMA transfer address is TIMER_MCHCTL2, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_MCH0CV</i>	DMA transfer address is TIMER_MCH0CV, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_MCH1CV</i>	DMA transfer address is TIMER_MCH1CV, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_MCH2CV</i>	DMA transfer address is TIMER_MCH2CV, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_MCH3CV</i>	DMA transfer address is TIMER_MCH3CV, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_CH0COMV_ADD</i>	DMA transfer address is TIMER_CH0COMV_ADD, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_CH1COMV_ADD</i>	DMA transfer address is TIMER_CH1COMV_ADD, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_CH2COMV_ADD</i>	DMA transfer address is TIMER_CH2COMV_ADD, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA TA_CH3COMV_ADD</i>	DMA transfer address is TIMER_CH3COMV_ADD, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i>	DMA transfer address is TIMER_CTL2, TIMERx(x=0,7)

<i>TA_CTL2</i>	
<i>TIMER_DMACFG_DMA</i> <i>TA_AFCTL</i>	DMA transfer address is <i>TIMER_AFCTL</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_IREP</i>	DMA transfer address is <i>TIMER_IREP</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_ADCRCTL</i>	DMA transfer address is <i>TIMER_ADCRCTL</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_ADCCR1</i>	DMA transfer address is <i>TIMER_ADCCR1</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_ADCCR2</i>	DMA transfer address is <i>TIMER_ADCCR2</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_ADCREP</i>	DMA transfer address is <i>TIMER_ADCREP</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_ADCTL</i>	DMA transfer address is <i>TIMER_ADCTL</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_SHUPM0</i>	DMA transfer address is <i>TIMER_SHUPM0</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_SHUPM1</i>	DMA transfer address is <i>TIMER_SHUPM1</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_CNT_FLOW_SEL</i>	DMA transfer address is <i>TIMER_CHCTL0</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<b>Input parameter{in}</b>	
<b>dma_lenth</b>	DMA transfer count
<i>TIMER_DMACFG_DMA</i> <i>TC_xTRANSFER</i>	( <i>x</i> =1~52), DMA transfer <i>x</i> 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 3-890. Function `timer_event_software_generate`**

<b>Function name</b>	<code>timer_event_software_generate</code>
<b>Function prototype</b>	<code>void timer_event_software_generate(uint32_t timer_periph, uint32_t event);</code>
<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=0,1,2,7)</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 generation, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_EVENT_SRC_CAPTURE_COMPARE_0</i>	channel 0 capture or compare event generation, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_EVENT_SRC_CAPTURE_COMPARE_1</i>	channel 1 capture or compare event generation, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_EVENT_SRC_CAPTURE_COMPARE_2</i>	channel 2 capture or compare event generation, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_EVENT_SRC_CAPTURE_COMPARE_3</i>	channel 3 capture or compare event generation, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_EVENT_SRC_COMMUTATION</i>	channel commutation event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_TRIGGER</i>	trigger event generation, <i>TIMERx(x=0..4,7)</i>
<i>TIMER_EVENT_SRC_BREAK</i>	break event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_MULTI_MODE_CAPTURE_COMPARE_0</i>	multi mode channel 0 capture or compare event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_MULTI_MODE_CAPTURE_COMPARE_1</i>	multi mode channel 1 capture or compare event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_MULTI_MODE_CAPTURE_COMPARE_2</i>	multi mode channel 2 capture or compare event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_MULTI_MODE_CAPTURE_COMPARE_3</i>	multi mode channel 3 capture or compare event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_ADDITIONAL_COMPARE_0</i>	channel 0 additional compare event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_ADDITIONAL_COMPARE_1</i>	channel 1 additional compare event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_ADDITIONAL_COMPARE_2</i>	channel 2 additional compare event generation, <i>TIMERx(x=0,7)</i>
<i>TIMER_EVENT_SRC_ADDITIONAL_COMPARE_3</i>	channel 3 additional compare event generation, <i>TIMERx(x=0,7)</i>
<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 3-891. Function timer\_break\_struct\_para\_init**

Function name	timer_break_struct_para_init
Function prototype	void timer_break_struct_para_init(timer_break_parameter_struct* breakpara);
Function descriptions	initialize the parameters of TIMER break parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	
breakpara	TIMER break parameter struct, the structure members can refer to <a href="#">Structure timer_break_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER break parameter struct with a default value */
```

```
timer_break_parameter_struct timer_breakpara;
```

```
timer_break_struct_para_init(&timer_breakpara);
```

### timer\_break\_config

The description of timer\_break\_config is shown as below:

**Table 3-892. Function timer\_break\_config**

Function name	timer_break_config
Function prototype	void timer_break_config(uint32_t timer_periph, timer_break_parameter_struct* breakpara);
Function descriptions	configure TIMER break function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	TIMER peripheral selection



Input parameter{in}	
<b>breakpara</b>	TIMER break parameter struct, the structure members can refer to <a href="#">Structure timer break parameter struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 break function */

timer_break_parameter_struct timer_breakpara;

timer_breakpara.runoffstate      = TIMER_ROS_STATE_DISABLE;
timer_breakpara.ideloffstate     = TIMER_IOS_STATE_DISABLE;
timer_breakpara.deadtime         = 0U;
timer_breakpara.outputautostate  = TIMER_OUTAUTO_ENABLE;
timer_breakpara.protectmode      = TIMER_CCHP_PROT_OFF;
timer_breakpara.breakstate       = TIMER_BREAK0_ENABLE;
timer_breakpara.breakfilter      = 0U;
timer_breakpara.breakpolarity    = TIMER_BREAK0_POLARITY_LOW;
timer_break_config(TIMER0, &timer_breakpara);

```

### timer\_break\_enable

The description of timer\_break\_enable is shown as below:

**Table 3-893. Function timer\_break\_enable**

<b>Function name</b>	timer_break_enable
<b>Function prototype</b>	void timer_break_enable(uint32_t timer_periph);
<b>Function descriptions</b>	enable 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>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<b>TIMERx(x=0,7)</b>	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable TIMER0 BREAK function*/

timer_break_enable(TIMER0);
```

### timer\_break\_disable

The description of timer\_break\_disable is shown as below:

**Table 3-894. Function timer\_break\_disable**

<b>Function name</b>	timer_break_disable
<b>Function prototype</b>	void timer_break_disable(uint32_t timer_periph);
<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)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TIMER0 BREAK function*/

timer_break_disable(TIMER0);
```

### timer\_automatic\_output\_enable

The description of timer\_automatic\_output\_enable is shown as below:

**Table 3-895. 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(x=0,7)</i>	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 3-896. 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(x=0,7)</i>	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 3-897. 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(x=0,7)</i>	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\_output\_struct\_para\_init

The description of timer\_channel\_output\_struct\_para\_init is shown as below:

**Table 3-898. Function timer\_channel\_output\_struct\_para\_init**

<b>Function name</b>	timer_channel_output_struct_para_init
<b>Function prototype</b>	void timer_channel_output_struct_para_init(timer_oc_parameter_struct* ocpa);
<b>Function descriptions</b>	initialize the parameters of TIMER channel output parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<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:

```
/* 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 3-899. Function timer\_channel\_output\_config**

<b>Function name</b>	timer_channel_output_config
<b>Function prototype</b>	void timer_channel_output_config(uint32_t timer_periph, uint16_t channel,

	timer_oc_parameter_struct* ocpara);
<b>Function descriptions</b>	configure TIMER 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=0,1,2,7)</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>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_ocintpara.outputstate = TIMER_CCX_ENABLE;

timer_ocintpara.outputnstate = TIMER_CCXN_ENABLE;

timer_ocintpara.ocpolarity = TIMER_OC_POLARITY_HIGH;

timer_ocintpara.ocnpolarity = TIMER_OCN_POLARITY_HIGH;

timer_ocintpara.ocidlestate = TIMER_OC_IDLE_STATE_HIGH;

timer_ocintpara.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 3-900. 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
<i>TIMERx(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<b>Input parameter{in}</b>	
<b>ocmode</b>	channel output compare mode
<i>TIMER_OC_MODE_TIMING</i>	timing mode (TIMERx(x=0,1,2,7))
<i>TIMER_OC_MODE_ACTIVE</i>	active mode (TIMERx(x=0,1,2,7))
<i>TIMER_OC_MODE_INACTIVE</i>	inactive mode (TIMERx(x=0,1,2,7))
<i>TIMER_OC_MODE_TOGGLE</i>	toggle mode (TIMERx(x=0,1,2,7))
<i>TIMER_OC_MODE_LOW</i>	force low mode (TIMERx(x=0,1,2,7))
<i>TIMER_OC_MODE_HIGH</i>	force high mode (TIMERx(x=0,1,2,7))
<i>TIMER_OC_MODE_PWM0</i>	PWM mode 0 (TIMERx(x=0,1,2,7))
<i>TIMER_OC_MODE_PWM1</i>	PWM mode 1 (TIMERx(x=0,1,2,7))
<i>TIMER_OC_MODE_COMPOSITE_PWM0</i>	COMPOSITE pwm mode0 (TIMERx(x=0,7))
<i>TIMER_OC_MODE_COMPOSITE_PWM1</i>	COMPOSITE pwm mode1 (TIMERx(x=0,7))
<i>TIMER_OC_MODE_ASYMMETRIC_PWM0</i>	ASYMMETRIC pwm mode0 (TIMERx(x=0,7))
<i>TIMER_OC_MODE_ASYMMETRIC_PWM1</i>	ASYMMETRIC pwm mode1 (TIMERx(x=0,7))
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel PWM 0 mode */
```

```
timer_channel_output_mode_config(TIMER0,TIMER_CH_0,TIMER_OC_MODE_PWM0);
```

### timer\_channel\_output\_pulse\_value\_config

The description of timer\_channel\_output\_pulse\_value\_config is shown as below:

**Table 3-901. 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
<i>TIMERx(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<b>Input parameter{in}</b>	
<b>pulse</b>	the integer part of channel output pulse value, 0~0xFFFF
<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\_shadow\_config

The description of timer\_channel\_output\_shadow\_config is shown as below:

**Table 3-902. 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(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<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\_compare\_fast\_config

The description of timer\_channel\_output\_compare\_fast\_config is shown as below:

**Table 3-903. Function timer\_channel\_output\_compare\_fast\_config**

<b>Function name</b>	timer_channel_output_compare_fast_config
<b>Function prototype</b>	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=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<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 channel 0 output compare fast function */
```

```
timer_channel_output_compare_fast_config(TIMER0, TIMER_CH_0, TIMER_OC_FAST_ENABLE);
```

## timer\_channel\_output\_clear\_config

The description of timer\_channel\_output\_clear\_config is shown as below:

**Table 3-904. 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>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER periphera
<i>TIMERx(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<b>Input parameter{in}</b>	
<b>occlear</b>	channel output clear function
<i>TIMER_OC_CLEAR_ENABLE</i>	channel output clear function enable
<i>TIMER_OC_CLEAR_DISABLE</i>	channel output clear function disable
<i>TIMER_OMC_CLEAR_ENABLE</i>	multi mode channel output clear function enable
<i>TIMER_OMC_CLEAR_DISABLE</i>	multi mode channel output clear function disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel 0 output clear function */
```

```
timer_channel_output_clear_config(TIMER0, TIMER_CH_0, TIMER_OC_CLEAR_ENABLE);
```

## timer\_channel\_output\_polarity\_config

The description of timer\_channel\_output\_polarity\_config is shown as below:

**Table 3-905. 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
<i>TIMERx(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<b>Input parameter{in}</b>	
<b>ocpolarity</b>	channel output polarity
<i>TIMER_OC_POLARITY_HIGH</i>	channel output polarity is high
<i>TIMER_OC_POLARITY_LOW</i>	channel output polarity is low
<i>TIMER_OMC_POLARITY_HIGH</i>	multi mode channel output polarity is high
<i>TIMER_OMC_POLARITY_LOW</i>	multi mode channel output polarity is low
<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 3-906. 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
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
<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>ocpolarity</b>	channel complementary output polarity
<i>TIMER_OCN_POLARITY_HIGH</i>	channel complementary output polarity is high
<i>TIMER_OCN_POLARITY_LOW</i>	channel complementary output polarity is low
<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 3-907. 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(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<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 3-908. 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)</i>	TIMER peripheral selection
<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>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 3-909. 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 3-910. Function timer\_input\_capture\_config**

<b>Function name</b>	timer_input_capture_config
<b>Function prototype</b>	void timer_input_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpara);
<b>Function descriptions</b>	configure TIMER input capture parameter
<b>Precondition</b>	-
<b>The called functions</b>	timer_channel_input_capture_prescaler_config
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<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:

```
/* configure TIMER0 input capture parameter */
```

```
timer_ic_parameter_struct timer_icinitpara;
```

```
timer_icinitpara.icpolarity = TIMER_IC_POLARITY_RISING;
```

```
timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTTI;
```

```

timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;

timer_icinitpara.icfilter      = 0x0;

timer_input_capture_config(TIMER0, TIMER_CH_0, &timer_icinitpara);

```

### timer\_channel\_input\_capture\_prescaler\_config

The description of timer\_channel\_input\_capture\_prescaler\_config is shown as below:

**Table 3-911. 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(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
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 3-912. Function timer\_channel\_capture\_value\_register\_read**

<b>Function name</b>	timer_channel_capture_value_register_read
<b>Function prototype</b>	uint32_t timer_channel_capture_value_register_read(uint32_t timer_periph, uint16_t channel);
<b>Function descriptions</b>	read TIMER channel capture compare register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</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=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	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 3-913. Function timer\_input\_pwm\_capture\_config**

<b>Function name</b>	timer_input_pwm_capture_config
<b>Function prototype</b>	void timer_input_pwm_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpwm);
<b>Function descriptions</b>	configure TIMER input pwm capture function
<b>Precondition</b>	-

<b>The called functions</b>	timer_channel_input_capture_prescaler_config
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<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
<b>Input parameter{in}</b>	
<b>icpwm</b>	TIMER channel input PWM 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:

```

/* 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 3-914. Function timer\_hall\_mode\_config**

<b>Function name</b>	timer_hall_mode_config
<b>Function prototype</b>	void timer_hall_mode_config(uint32_t timer_periph, uint32_t hallmode);
<b>Function descriptions</b>	configure TIMER hall sensor mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>hallmode</b>	TIMER hall sensor mode state
<i>TIMER_HALLINTERFA</i>	TIMER hall sensor mode enable

<i>CE_ENABLE</i>	
<i>TIMER_HALLINTERFA</i> <i>CE_DISABLE</i>	TIMER hall sensor mode disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-915. 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>	-
<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:

```
/* 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 3-916. 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)</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
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3
<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 3-917. 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>	-

Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
Input parameter{in}	
<b>channel</b>	TIMER channel
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3
Input parameter{in}	
<b>multi_mode_sel</b>	multi mode channel mode selection
<i>TIMER_MCH_MODE_INDEPENDENTLY</i>	multi mode channel work in independently mode
<i>TIMER_MCH_MODE_COMPLEMENTARY</i>	multi mode channel work in complementary output mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select TIMER0 multi mode channel 0 mode */
```

```
timer_multi_mode_channel_mode_config (TIMER0, TIMER_MCH_0, TIMER_MCH_MODE_INDEPENDENTLY);
```

### timer\_input\_trigger\_source\_select

The description of timer\_input\_trigger\_source\_select is shown as below:

**Table 3-918. 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>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	please refer to the following parameters
Input parameter{in}	
<b>intrigger</b>	input trigger source
<i>TIMER_SMCFG_TRGS_EL_NONE</i>	event mode disable
<i>TIMER_SMCFG_TRGS</i>	internal trigger 0

<i>EL_ITI0</i>	
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI1</i>	internal trigger 1
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI2</i>	internal trigger 2
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI3</i>	internal trigger 3
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOF_ED</i>	TIO edge detector
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOFE0</i>	filtered channel 0 input
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI1FE1</i>	filtered channel 1 input
<i>TIMER_SMCFG_TRGS</i> <i>EL_ETIFP</i>	filtered external trigger input
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI2FE2</i>	filtered channel 2 input(TIMERx(x=0,7))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI3FE3</i>	filtered channel 3 input(TIMERx(x=0,7))
<i>TIMER_SMCFG_TRGS</i> <i>EL_MCI0FEM0</i>	filtered multi mode channel 0 input(TIMERx(x=0,7))
<i>TIMER_SMCFG_TRGS</i> <i>EL_MCI1FEM1</i>	filtered multi mode channel 1 input(TIMERx(x=0,7))
<i>TIMER_SMCFG_TRGS</i> <i>EL_MCI2FEM2</i>	filtered multi mode channel 2 input(TIMERx(x=0,7))
<i>TIMER_SMCFG_TRGS</i> <i>EL_MCI3FEM3</i>	filtered multi mode channel 3 input(TIMERx(x=0,7))
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0 input trigger source */
```

```
timer_input_trigger_source_select (TIMER0, TIMER_SMCFG_TRGSEL_ITI0);
```

### timer\_master\_output\_trigger\_source\_select

The description of timer\_master\_output\_trigger\_source\_select is shown as below:

**Table 3-919. Function timer\_master\_output\_trigger\_source\_select**

<b>Function name</b>	timer_master_output_trigger_source_select
<b>Function prototype</b>	void timer_master_output0_trigger_source_select(uint32_t timer_periph, uint32_t outtrigger);

<b>Function descriptions</b>	select TIMER master mode output trigger source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>outrigger</b>	trigger output source
<i>TIMER_TRI_OUT_SRC_RESET</i>	the UPG bit as trigger output
<i>TIMER_TRI_OUT_SRC_ENABLE</i>	the counter enable signal <i>TIMER_CTL0_CEN</i> as trigger output
<i>TIMER_TRI_OUT_SRC_UPDATE</i>	update event as trigger output
<i>TIMER_TRI_OUT_SRC_CH0</i>	a capture or a compare match occurred in channel0 as trigger output TRGO
<i>TIMER_TRI_OUT_SRC_O0CPRE</i>	O0CPRE as trigger output
<i>TIMER_TRI_OUT_SRC_O1CPRE</i>	O1CPRE as trigger output
<i>TIMER_TRI_OUT_SRC_O2CPRE</i>	O2CPRE as trigger output
<i>TIMER_TRI_OUT_SRC_O3CPRE</i>	O3CPRE as trigger output
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0 master mode output trigger source */
```

```
timer_master_output_trigger_source_select (TIMER0, TIMER_TRI_OUT0_SRC_RESET);
```

### timer\_slave\_mode\_select

The description of timer\_slave\_mode\_select is shown as below:

**Table 3-920. Function timer\_slave\_mode\_select**

<b>Function name</b>	timer_slave_mode_select
<b>Function prototype</b>	void timer_slave_mode_select(uint32_t timer_periph, uint32_t slavemode);
<b>Function descriptions</b>	select TIMER slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>slavemode</b>	slave mode
<i>TIMER_SLAVE_MODE_DISABLE</i>	slave mode disable
<i>TIMER_ENCODER_MODE_DE0</i>	quadrature decoder mode 0
<i>TIMER_ENCODER_MODE_DE1</i>	quadrature decoder mode 1
<i>TIMER_ENCODER_MODE_DE2</i>	quadrature decoder mode 2
<i>TIMER_SLAVE_MODE_RESTART</i>	restart mode
<i>TIMER_SLAVE_MODE_PAUSE</i>	pause mode
<i>TIMER_SLAVE_MODE_EVENT</i>	event mode
<i>TIMER_SLAVE_MODE_EXTERNAL0</i>	external clock mode 0
<i>TIMER_SLAVE_MODE_RESTART_EVENT</i>	restart + event mode
<i>TIMER_NONQUAD_MODE_DE0</i>	non-quadrature decoder mode 0
<i>TIMER_NONQUAD_MODE_DE1</i>	non-quadrature decoder mode 1
<i>TIMER_NONQUAD_MODE_DE2</i>	non-quadrature decoder mode 2
<i>TIMER_NONQUAD_MODE_DE3</i>	non-quadrature decoder mode 3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0 slave mode */
```

```
timer_slave_mode_select(TIMER0, TIMER_ENCODER_MODE0);
```

### timer\_master\_slave\_mode\_config

The description of timer\_master\_slave\_mode\_config is shown as below:



**Table 3-921. Function timer\_master\_slave\_mode\_config**

<b>Function name</b>	timer_master_slave_mode_config
<b>Function prototype</b>	void timer_master_slave_mode_config(uint32_t timer_periph, uint32_t masterslave);
<b>Function descriptions</b>	configure TIMER master slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>masterslave</b>	master slave mode state
<i>TIMER_MASTER_SLAVE_MODE_ENABLE</i>	master slave mode enable
<i>TIMER_MASTER_SLAVE_MODE_DISABLE</i>	master slave mode disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-922. Function timer\_external\_trigger\_config**

<b>Function name</b>	timer_external_trigger_config
<b>Function prototype</b>	void timer_external_trigger_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
<b>Function descriptions</b>	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=0,1,2,7)</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>DIV2</i>	
<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 3-923. 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=0,1,2,7)</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
<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 3-924. 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=0,1,2,7)</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>DE1</i>	
<i>TIMER_DECODER_MO</i> <i>DE2</i>	decoder mode 2
<i>TIMER_DECODER_MO</i> <i>DE3</i>	decoder mode 3
<b>Input parameter{in}</b>	
<b>ic0polarity</b>	CI0 input capture polarity
<i>TIMER_IC_POLARITY_</i> <i>RISING</i>	capture rising edge
<i>TIMER_IC_POLARITY_</i> <i>FALLING</i>	capture falling edge
<i>TIMER_IC_POLARITY_</i> <i>BOTH_EDGE</i>	active both edge
<b>Input parameter{in}</b>	
<b>ic1polarity</b>	CI1 input capture polarity
<i>TIMER_IC_POLARITY_</i> <i>RISING</i>	capture rising edge
<i>TIMER_IC_POLARITY_</i> <i>FALLING</i>	capture falling edge
<i>TIMER_IC_POLARITY_</i> <i>BOTH_EDGE</i>	active both edge
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 non-quadrature decoder mode */
```

```
timer_decoder_mode_config (TIMER0, TIMER_NONQUAD_MODE0, TIMER_IC_POLARITY_RISING, TIMER_IC_POLARITY_RISING);
```

### timer\_internal\_clock\_config

The description of timer\_internal\_clock\_config is shown as below:

**Table 3-925. 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=0,1,2,7)</i>	TIMER peripheral selection

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 internal clock mode */
```

```
timer_internal_clock_config (TIMER0);
```

### timer\_internal\_trigger\_as\_external\_clock\_config

The description of timer\_internal\_trigger\_as\_external\_clock\_config is shown as below:

**Table 3-926. Function timer\_internal\_trigger\_as\_external\_clock\_config**

Function name	timer_internal_trigger_as_external_clock_config
Function prototype	void timer_internal_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t intrigger);
Function descriptions	configure TIMER the internal trigger as external clock input
Precondition	-
The called functions	timer_input_trigger_source_select
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
Input parameter{in}	
<b>intrigger</b>	trigger selection
<i>TIMER_SMCFG_TRGS EL_ITI0</i>	internal trigger input 0 (ITI0)
<i>TIMER_SMCFG_TRGS EL_ITI1</i>	internal trigger input 1 (ITI1)
<i>TIMER_SMCFG_TRGS EL_ITI2</i>	internal trigger input 2 (ITI2)
<i>TIMER_SMCFG_TRGS EL_ITI3</i>	internal trigger input 3 (ITI3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 the internal trigger ITI0 as external clock input */
```

```
timer_internal_trigger_as_external_clock_config(TIMER0, TIMER_SMCFG_TRGSEL_ITI0);
```

## timer\_external\_trigger\_as\_external\_clock\_config

The description of timer\_external\_trigger\_as\_external\_clock\_config is shown as below:

**Table 3-927. 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=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>extrigger</b>	external trigger selection
<i>TIMER_SMCFG_TRGS EL_CIOF_ED</i>	TIO edge detector
<i>TIMER_SMCFG_TRGS EL_CIOFE0</i>	filtered channel 0 input
<i>TIMER_SMCFG_TRGS EL_CI1FE1</i>	filtered channel 1 input
<b>Input parameter{in}</b>	
<b>expolarity</b>	external trigger polarity
<i>TIMER_IC_POLARITY_ RISING</i>	active high or rising edge active
<i>TIMER_IC_POLARITY_ FALLING</i>	active low or falling edge active
<i>TIMER_IC_POLARITY_ BOTH_EDGE</i>	active both edge
<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_CIO  
FE0, TIMER_IC_POLARITY_RISING, 0);
```

## timer\_slave\_mode\_input\_config

The description of timer\_slave\_mode\_input\_config is shown as below:

**Table 3-928. Function timer\_slave\_mode\_input\_config**

<b>Function name</b>	timer_slave_mode_input_config
<b>Function prototype</b>	void timer_slave_mode_input_config(uint32_t timer_periph, uint32_t channel, uint16_t extpolarity, uint32_t extfilter)
<b>Function descriptions</b>	configure TIMER slave mode input
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	external trigger selection
<i>TIMER_CH_0</i>	TIMER channel 0 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_2</i>	TIMER channel 2 (TIMERx(x=0,1,2,7))
<i>TIMER_CH_3</i>	TIMER channel 3 (TIMERx(x=0,1,2,7))
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0 (TIMERx(x=0,7))
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1 (TIMERx(x=0,7))
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2 (TIMERx(x=0,7))
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3 (TIMERx(x=0,7))
<b>Input parameter{in}</b>	
<b>expolarity</b>	external trigger polarity
<i>TIMER_IC_POLARITY_RISING</i>	active high or rising edge active
<i>TIMER_IC_POLARITY_FALLING</i>	active low or falling edge active
<i>TIMER_IC_POLARITY_BOTH_EDGE</i>	active both edge
<i>TIMER_IMC_POLARITY_RISING</i>	active high or rising edge active
<i>TIMER_IMC_POLARITY_FALLING</i>	active low or falling edge active
<i>TIMER_IMC_POLARITY_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 CH0 slave mode input polarity */
```

```
timer_slave_mode_input_config (TIMER0, TIMER_CH_0, TIMER_IC_POLARITY_RISING,  
0);
```

### timer\_external\_clock\_mode0\_config

The description of timer\_external\_clock\_mode0\_config is shown as below:

**Table 3-929. Function timer\_external\_clock\_mode0\_config**

Function name	timer_external_clock_mode0_config
Function prototype	void timer_external_clock_mode0_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER the external clock mode0
Precondition	-
The called functions	timer_external_trigger_config
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,1,2,7)	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 3-930. 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
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<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
<b>Input parameter{in}</b>	
<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
<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 3-931. 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
<i>TIMERx(x=0,1,2,7)</i>	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 3-932. Function timer\_write\_chxval\_register\_config**

<b>Function name</b>	timer_write_chxval_register_config
<b>Function prototype</b>	void timer_write_chxval_register_config(uint32_t timer_periph, uint16_t ccsel);
<b>Function descriptions</b>	configure TIMER write CHxVAL register selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,1,2,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>ccsel</b>	write CHxVAL register selection
<i>TIMER_CHVSEL_DISABLE</i>	no effect
<i>TIMER_CHVSEL_ENABLE</i>	when write the CHxVAL register, if the write value is same as the CHxVAL value, the write access is ignored
<b>Output parameter{out}</b>	
-	-

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 3-933. 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(x=0,7)</i>	TIMER peripheral selection
Input parameter{in}	
<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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER output value selection */
```

```
timer_output_value_selection_config(TIMER0, TIMER_OUTSEL_ENABLE);
```

### timer\_output\_match\_pulse\_select

The description of timer\_output\_match\_pulse\_select is shown as below:

**Table 3-934. Function timer\_output\_match\_pulse\_select**

Function name	timer_output_match_pulse_select
Function prototype	void timer_output_match_pulse_select(uint32_t timer_periph, uint32_t channel, uint16_t pulsesel);

<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
<i>TIMERx(x=0,7)</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>pulsesel</b>	output match pulse selection
<i>TIMER_PULSE_OUTPUT_T_NORMAL</i>	channel output normal
<i>TIMER_PULSE_OUTPUT_T_CNT_UP</i>	pulse output only when counting up
<i>TIMER_PULSE_OUTPUT_T_CNT_DOWN</i>	pulse output only when counting down
<i>TIMER_PULSE_OUTPUT_T_CNT_BOTH</i>	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\_pulse\_value\_config

The description of timer\_channel\_composite\_pwm\_pulse\_config is shown as below:

**Table 3-935. Function timer\_channel\_composite\_pwm\_pulse\_config**

<b>Function name</b>	timer_channel_composite_pwm_pulse_config
<b>Function prototype</b>	void timer_channel_composite_pwm_pulse_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=0,7)</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>pulse</b>	channel compare value(0~0xFFFF)
<b>Input parameter{in}</b>	
<b>add_pulse</b>	channel additional compare value(0~0xFFFF)
<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\_asymmetric\_pwm\_pulse\_value\_config

The description of timer\_channel\_asymmetric\_pwm\_pulse\_config is shown as below:

**Table 3-936. Function timer\_channel\_asymmetric\_pwm\_pulse\_config**

<b>Function name</b>	timer_channel_asymmetric_pwm_pulse_config
<b>Function prototype</b>	void timer_channel_asymmetric_pwm_pulse_config (uint32_t timer_periph, uint32_t channel, uint32_t pulse, uint32_t add_pulse);
<b>Function descriptions</b>	configure the TIMER asymmetric 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=0,7)</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>pulse</b>	channel compare value(0~0xFFFF)

Input parameter{in}	
<b>add_pulse</b>	channel additional compare value(0~0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 output pulse value */
```

```
timer_channel_asymmetric_pwm_pulse_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 3-937. 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>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
Input parameter{in}	
<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
Input parameter{in}	
<b>value</b>	channel additional compare value(0~0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 channel 0 additional compare value */
```

```
timer_channel_additional_compare_value_config (TIMER0, TIMER_CH_0, 399);
```

## timer\_channel\_additional\_compare\_value\_read

The description of timer\_channel\_additional\_compare\_value\_read is shown as below:

**Table 3-938. Function timer\_channel\_additional\_compare\_value\_read**

<b>Function name</b>	timer_channel_additional_compare_value_read
<b>Function prototype</b>	uint16_t timer_channel_additional_compare_value_read(uint32_t timer_periph, uint16_t channel);
<b>Function descriptions</b>	read 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=0,7)</i>	TIMER peripheral selection
<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>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	channel additional compare value, 0~0xFFFF

Example:

```
/* get TIMER autoreload register value */

uint16_t i = 0;

i = timer_channel_additional_compare_value_read(TIMER0, TIMER_CH_0);
```

## timer\_channel\_additional\_output\_shadow\_config

The description of timer\_channel\_additional\_output\_shadow\_config is shown as below:

**Table 3-939. 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=0,7)</i>	please refer to the following parameters

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

Example:

```
/*configure TIMER0 channel 0 additional output shadow function */
```

```
timer_channel_additional_output_shadow_config (TIMER0, TIMER_CH_0, TIMER_OC_SHADOW_ENABLE);
```

### timer\_break\_external\_input\_enable

The description of timer\_break\_external\_input\_enable is shown as below:

**Table 3-940. Function timer\_break\_external\_input\_enable**

<b>Function name</b>	timer_break_external_input_enable
<b>Function prototype</b>	void timer_break_external_input_enable(uint32_t timer_periph)
<b>Function descriptions</b>	enable break external input
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMER break enternal input */
```

```
timer_break_external_input_enable (TIMER0);
```



## timer\_break\_cmp\_enable

The description of timer\_break\_cmp\_enable is shown as below:

**Table 3-941. Function timer\_break\_cmp\_enable**

<b>Function name</b>	timer_break_cmp_enable
<b>Function prototype</b>	void timer_break_cmp_enable(uint32_t timer_periph, uint32_t break_cmp)
<b>Function descriptions</b>	enable break cmp
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>break_cmp</b>	Break function CMP selection
<i>TIMER_BREAK_CMPX</i> (X=0..3)	CMP0/1/2/3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the TIMER break cmp input */
```

```
timer_break_cmp_enable (TIMER0, TIMER_BREAK_CMP0);
```

## timer\_break\_external\_input\_disable

The description of timer\_break\_external\_input\_disable is shown as below:

**Table 3-942. Function timer\_break\_external\_input\_disable**

<b>Function name</b>	timer_break_external_input_disable
<b>Function prototype</b>	void timer_break_external_input_disable(uint32_t timer_periph)
<b>Function descriptions</b>	disable break external input
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the TIMER break enternal input */
```

```
timer_break_external_input_disable (TIMER0);
```

### timer\_break\_cmp\_disable

The description of timer\_break\_cmp\_disable is shown as below:

**Table 3-943. Function timer\_break\_cmp\_disable**

<b>Function name</b>	timer_break_cmp_disable
<b>Function prototype</b>	void timer_break_cmp_disable(uint32_t timer_periph, uint32_t break_cmp)
<b>Function descriptions</b>	disable break cmp
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>break_cmp</b>	Break function CMP selection
<i>TIMER_BREAK_CMPX</i> (X=0..3)	CMP0/1/2/3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the TIMER break cmp input */
```

```
timer_break_cmp_disable (TIMER0, TIMER_BERAK_CMP0);
```

### timer\_break\_external\_input\_polarity\_config

The description of timer\_break\_external\_input\_polarity\_config is shown as below:

**Table 3-944. Function timer\_break\_external\_input\_polarity\_config**

<b>Function name</b>	timer_break_external_input_polarity_config
<b>Function prototype</b>	void timer_break_external_input_polarity_config(uint32_t timer_periph, uint32_t polarity);
<b>Function descriptions</b>	configure TIMER break input 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)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	

<b>bkinpolarity</b>	break external input polarity
<i>TIMER_BRKIN_POLARITY_NONINVERTED</i>	input signal will not be inverted
<i>TIMER_BRKIN_POLARITY_INVERTED</i>	input signal will be inverted
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER break external input polarity */
```

```
timer_break_external_input_polarity_config (TIMER0, TIMER_BRKIN_POLARITY_NONINVERTED);
```

### timer\_break\_cmp\_polarity\_config

The description of timer\_break\_cmp\_polarity\_config is shown as below:

**Table 3-945. Function timer\_break\_cmp\_polarity\_config**

<b>Function name</b>	timer_break_cmp_polarity_config
<b>Function prototype</b>	void timer_break_cmp_polarity_config(uint32_t timer_periph, uint32_t break_cmp, uint32_t polarity)
<b>Function descriptions</b>	configure TIMER break cmp 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)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>break_cmp</b>	break function CMP selection
<i>TIMER_BREAK_CMPX (X=0..3)</i>	CMP0/1/2/3
<b>Input parameter{in}</b>	
<b>polarity</b>	break cmp polarity
<i>TIMER_BRKIN_CMP_INVERTED</i>	break cmp polarity will be inverted
<i>TIMER_BRKIN_CMP_NONINVERTED</i>	break cmp polarity will not be inverted
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER break cmp polarity */
```

```
timer_break_cmp_polarity_config (TIMER0, TIMER_BRKIN_CMP0, TIMER_BRKIN_CMP_
NONINVERTED);
```

### timer\_break\_auto\_recover\_event\_select

The description of timer\_break\_auto\_recover\_event\_select is shown as below:

**Table 3-946. Function timer\_break\_auto\_recover\_event\_select**

<b>Function name</b>	timer_break_auto_recover_event_select
<b>Function prototype</b>	void timer_break_auto_recover_event_select(uint32_t timer_periph, uint32_t event)
<b>Function descriptions</b>	select break auto recover 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)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>event</b>	break auto recovery event
<i>TIMER_BRKAU_MODE_FLOW</i>	break recover output by next flow
<i>TIMER_BRKAU_MODE_OVERFLOW</i>	break recover output by next overflow
<i>TIMER_BRKAU_MODE_UNDERFLOW</i>	break recover output by next underflow
<i>TIMER_BRKAU_MODE_UPADATE_EVENT</i>	break recover output by next update event
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 break auto recover event selection */
```

```
timer_break_auto_recover_event_select (TIMER0, TIMER_BRKAU_MODE_FLOW);
```

### timer\_trigger\_adc\_compare\_enable

The description of timer\_trigger\_adc\_compare\_enable is shown as below:

**Table 3-947. Function timer\_trigger\_adc\_compare\_enable**

<b>Function name</b>	timer_trigger_adc_compare_enable
----------------------	----------------------------------

<b>Function prototype</b>	void timer_trigger_adc_compare_enable(uint32_t timer_periph, uint32_t compare_time)
<b>Function descriptions</b>	enable timer compare event produce a ADC converter trigger signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>compare_time</b>	ADC converter trigger signal compare time
<i>TIMER_ADCRCTL_ADT2DWGTEN</i>	ADC converter trigger signal (TIMERx_TRGB) enabled during CNT down-count operation
<i>TIMER_ADCRCTL_ADT2UPGTEN</i>	ADC converter trigger signal (TIMERx_TRGB) enabled during CNT up-count operation
<i>TIMER_ADCRCTL_ADT1DWGTEN</i>	ADC converter trigger signal (TIMERx_TRGA) enabled during CNT down-count operation
<i>TIMER_ADCRCTL_ADT1UPGTEN</i>	ADC converter trigger signal (TIMERx_TRGA) enabled during CNT up-count operation
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TIMER0 compare event produce a ADC converter trigger signal */
timer_trigger_adc_compare_enable (TIMER0, TIMER_ADCRCTL_ADT2DWGTEN);
```

### timer\_trigger\_adc\_compare\_disable

The description of timer\_trigger\_adc\_compare\_disable is shown as below:

**Table 3-948. Function timer\_trigger\_adc\_compare\_disable**

<b>Function name</b>	timer_trigger_adc_compare_disable
<b>Function prototype</b>	void timer_trigger_adc_compare_disable(uint32_t timer_periph, uint32_t compare_time)
<b>Function descriptions</b>	disable timer compare event produce a ADC converter trigger signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>compare_time</b>	ADC converter trigger signal compare time
<i>TIMER_ADCRCTL_ADT</i>	ADC converter trigger signal (TIMERx_TRGB) enabled during CNT down-

<i>2DWGTEN</i>	count operation
<i>TIMER_ADCRCTL_ADT2UPGTEN</i>	ADC converter trigger signal (TIMERx_TRGB) enabled during CNT up-count operation
<i>TIMER_ADCRCTL_ADT1DWGTEN</i>	ADC converter trigger signal (TIMERx_TRGA) enabled during CNT down-count operation
<i>TIMER_ADCRCTL_ADT1UPGTEN</i>	ADC converter trigger signal (TIMERx_TRGA) enabled during CNT up-count operation
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TIMER0 compare event produce a ADC converter trigger signal */
```

```
timer_trigger_adc_compare_disable (TIMER0, TIMER_ADCRCTL_ADT2DWGTEN);
```

### timer\_trigger\_adc\_flow\_enable

The description of timer\_trigger\_adc\_flow\_enable is shown as below:

**Table 3-949. Function timer\_trigger\_adc\_flow\_enable**

<b>Function name</b>	timer_trigger_adc_flow_enable
<b>Function prototype</b>	void timer_trigger_adc_flow_enable(uint32_t timer_periph, uint32_t flow_timer)
<b>Function descriptions</b>	enable timer flow event produce a ADC converter trigger signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>flow_timer</b>	TIMER trigger ADC flow signal
<i>TIMER_ADCTL_ADTSUF</i>	select underflow signal
<i>TIMER_ADCTL_ADTSOF</i>	select overflow signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TIMER0 flow event produce a ADC converter trigger signal */
```

timer\_trigger\_adc\_flow\_enable (TIMER0, TIMER\_ADCTL\_ADTSUF);

### timer\_trigger\_adc\_flow\_disable

The description of timer\_trigger\_adc\_flow\_disable is shown as below:

**Table 3-950. Function timer\_trigger\_adc\_flow\_disable**

<b>Function name</b>	timer_trigger_adc_flow_disable
<b>Function prototype</b>	void timer_trigger_adc_flow_disable(uint32_t timer_periph, uint32_t flow_timer)
<b>Function descriptions</b>	disable timer flow event produce a ADC converter trigger signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>flow_timer</b>	TIMER trigger ADC flow signal
<i>TIMER_ADCTL_ADTSUF</i> <i>F</i>	select underflow signal
<i>TIMER_ADCTL_ADTSO</i> <i>F</i>	select overflow signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TIMER0 flow event produce a ADC converter trigger signal */
```

```
timer_trigger_adc_flow_disable (TIMER0, TIMER_ADCTL_ADTSUF);
```

### timer\_trigger\_adc\_compare\_value\_config

The description of timer\_trigger\_adc\_compare\_value\_config is shown as below:

**Table 3-951. Function timer\_trigger\_adc\_compare\_value\_config**

<b>Function name</b>	timer_trigger_adc_compare_value_config
<b>Function prototype</b>	void timer_trigger_adc_compare_value_config(uint32_t timer_periph, uint32_t compare, uint16_t value);
<b>Function descriptions</b>	configure adc trigger compare value 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)</i>	please refer to the following parameters

Input parameter{in}	
<b>compare</b>	ADC trigger compare register
<i>TIMER_ADC_COMPAR_E1</i>	select the <i>TIMER_ADC_CCR1</i> register
<i>TIMER_ADC_COMPAR_E2</i>	select the <i>TIMER_ADC_CCR2</i> register
Input parameter{in}	
<b>value</b>	compare value (0~0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 trigger ADC compare register value */
```

```
timer_trigger_adc_compare_value_config (TIMER0, TIMER_ADC_COMPARE1, 399);
```

### timer\_trigger\_adc\_repetition\_value\_config

The description of `timer_trigger_adc_repetition_value_config` is shown as below:

**Table 3-952. Function `timer_trigger_adc_repetition_value_config`**

<b>Function name</b>	<code>timer_trigger_adc_repetition_value_config</code>
<b>Function prototype</b>	<code>void timer_trigger_adc_repetition_value_config(uint32_t timer_periph, uint32_t timer_adc_crep, uint16_t value);</code>
<b>Function descriptions</b>	configure the TIMER trigger ADC repetition register value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
Input parameter{in}	
<b>timer_adc_crep</b>	TIMER trigger ADC repetition register
<i>TIMER_ADC_REPA</i>	select the <i>ADC_REP_A</i> register
<i>TIMER_ADC_REPB</i>	select the <i>ADC_REP_B</i> register
<i>TIMER_ADC_REPAB</i>	select the <i>ADC_REP_AB</i> register
Input parameter{in}	
<b>value</b>	compare value (0~0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:



```
/* configure TIMER0 trigger ADC repetition value */
```

```
timer_trigger_adc_repetition_value_config (TIMER0, TIMER_ADC_REPA, 1);
```

### timer\_trigger\_adc\_repetition\_decrement\_select

The description of timer\_trigger\_adc\_repetition\_decrement\_select is shown as below:

**Table 3-953. Function timer\_trigger\_adc\_repetition\_decrement\_select**

<b>Function name</b>	timer_trigger_adc_repetition_decrement_select
<b>Function prototype</b>	void timer_trigger_adc_repetition_decrement_select(uint32_t timer_periph, uint32_t timer_adc_crep, uint16_t source)
<b>Function descriptions</b>	select adc repetition_decrement 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)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>timer_adc_crep</b>	TIMER trigger ADC repetition register
<i>TIMER_ADC_REPA</i>	select the ADC_REPA register
<i>TIMER_ADC_REPB</i>	select the ADC_REPB register
<b>Input parameter{in}</b>	
<b>source</b>	decrement source
<i>TIMER_ADC_REP_DEC_OVERFLOW</i>	ADC repetition decrement when count overflow
<i>TIMER_ADC_REP_DEC_UNDERFLOW</i>	ADC repetition decrement when count underflow
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0 trigger ADC repetition_decrement source */
```

```
timer_trigger_adc_repetition_decrement_select (TIMER0, TIMER_ADC_REPA, TIMER_ADC_REP_DEC_OVERFLOW);
```

### timer\_trigger\_adc\_repetition\_value\_reload

The description of timer\_trigger\_adc\_repetition\_value\_reload is shown as below:

**Table 3-954. timer\_trigger\_adc\_repetition\_value\_reload**

<b>Function name</b>	timer_trigger_adc_repetition_value_reload
<b>Function prototype</b>	void timer_trigger_adc_repetition_value_reload(uint32_t timer_periph);

<b>Function descriptions</b>	reload TIMER trigger ADC 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)</i>	please refer to the following parameters
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reload TIMER0 trigger ADC repetition value */
```

```
timer_trigger_adc_repetition_value_reload (TIMER0);
```

### timer\_trigger\_adc\_compare\_value\_shadow\_enable

The description of timer\_trigger\_adc\_compare\_value\_shadow\_enable is shown as below:

**Table 3-955. Function timer\_trigger\_adc\_compare\_value\_shadow\_enable**

<b>Function name</b>	timer_trigger_adc_compare_value_shadow_enable
<b>Function prototype</b>	void timer_trigger_adc_compare_value_shadow_enable(uint32_t timer_periph, uint32_t timer_adc_cmpval);
<b>Function descriptions</b>	enable ADC compare register shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>timer_adc_cmpval</b>	ADC compare shadow register
<i>TIMER_ADCTL_ADTPREEN1</i>	select the TIMER_ADCCR1 shadow register
<i>TIMER_ADCTL_ADTPREEN2</i>	select the TIMER_ADCCR2 shadow register
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TIMER0 trigger ADC compare register shadow function */
```

```
timer_trigger_adc_compare_value_shadow_enable (TIMER0, TIMER_ADCTL_ADTPREE
```

N1);

### timer\_trigger\_adc\_compare\_value\_shadow\_disable

The description of timer\_trigger\_adc\_compare\_value\_shadow\_disable is shown as below:

**Table 3-956. Function timer\_trigger\_adc\_compare\_value\_shadow\_disable**

<b>Function name</b>	timer_trigger_adc_compare_value_shadow_disable
<b>Function prototype</b>	void timer_trigger_adc_compare_value_shadow_disable(uint32_t timer_periph, uint32_t timer_adc_cmpval);
<b>Function descriptions</b>	disable ADC compare register shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>timer_adc_cmpval</b>	ADC compare shadow register
<i>TIMER_ADCTL_ADTP REEN1</i>	select the TIMER_ADCCR1 shadow register
<i>TIMER_ADCTL_ADTP REEN2</i>	select the TIMER_ADCCR2 shadow register
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TIMER0 trigger ADC compare register shadow function */
```

```
timer_trigger_adc_compare_value_shadow_disable (TIMER0, TIMER_ADCTL_ADTPREEN1);
```

### timer\_trigger\_adc\_monitor\_config

The description of timer\_trigger\_adc\_monitor\_config is shown as below:

**Table 3-957. Function timer\_trigger\_adc\_monitor\_config**

<b>Function name</b>	timer_trigger_adc_monitor_config
<b>Function prototype</b>	void timer_trigger_adc_monitor_config(uint32_t timer_periph, uint32_t adcsel, uint32_t adcsel_sel, ControlStatus newvalue);
<b>Function descriptions</b>	configure ADC trigger signal monitoring 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)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>adcsn</b>	ADC signal monitor
<i>SYSCFG_TADSRCFG _ADCSM1</i>	ADC SM1 pin output
<i>SYSCFG_TADSRCFG _ADCSM2</i>	ADC SM2 pin output
<b>Input parameter{in}</b>	
<b>adcsn_sel</b>	source of monitor
<i>SYSCFG_ADCSM1_TI MER0_TRGOF</i>	source of monitor is TIMER0_TRGOF
<i>SYSCFG_ADCSM1_TI MER0_TRGUF</i>	source of monitor is TIMER0_TRGUF
<i>SYSCFG_ADCSM1_TI MER0_TRGA</i>	source of monitor is TIMER0_TRGA
<i>SYSCFG_ADCSM1_TI MER0_TRGB</i>	source of monitor is TIMER0_TRGB
<i>SYSCFG_ADCSM1_TI MER0_TRGAB</i>	source of monitor is TIMER0_TRGAB
<i>SYSCFG_ADCSM1_TI MER7_TRGOF</i>	source of monitor is TIMER7_TRGOF
<i>SYSCFG_ADCSM1_TI MER7_TRGUF</i>	source of monitor is TIMER7_TRGUF
<i>SYSCFG_ADCSM1_TI MER7_TRGA</i>	source of monitor is TIMER7_TRGA
<i>SYSCFG_ADCSM1_TI MER7_TRGB</i>	source of monitor is TIMER7_TRGB
<i>SYSCFG_ADCSM1_TI MER7_TRGAB</i>	source of monitor is TIMER7_TRGAB
<i>SYSCFG_ADCSM2_TI MER0_TRGOF</i>	source of monitor is TIMER0_TRGOF
<i>SYSCFG_ADCSM2_TI MER0_TRGUF</i>	source of monitor is TIMER0_TRGUF
<i>SYSCFG_ADCSM2_TI MER0_TRGA</i>	source of monitor is TIMER0_TRGA
<i>SYSCFG_ADCSM2_TI MER0_TRGB</i>	source of monitor is TIMER0_TRGB
<i>SYSCFG_ADCSM2_TI MER0_TRGAB</i>	source of monitor is TIMER0_TRGAB
<i>SYSCFG_ADCSM2_TI MER7_TRGOF</i>	source of monitor is TIMER7_TRGOF
<i>SYSCFG_ADCSM2_TI MER7_TRGUF</i>	source of monitor is TIMER7_TRGUF

<i>SYSCFG_ADCSM2_TIMER7_TRGA</i>	source of monitor is TIMER7_TRGA
<i>SYSCFG_ADCSM2_TIMER7_TRGB</i>	source of monitor is TIMER7_TRGB
<i>SYSCFG_ADCSM2_TIMER7_TRGAB</i>	source of monitor is TIMER7_TRGAB
<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 trigger signal monitoring function */
```

```
timer_trigger_adc_monitor_config (TIMER0, SYSCFG_ADCSM1_TIMER0_TRGOF, ENABLE);
```

### timer\_register\_update\_event\_select

The description of timer\_register\_update\_event\_select is shown as below:

**Table 3-958. Function timer\_register\_update\_event\_select**

<b>Function name</b>	timer_register_update_event_select
<b>Function prototype</b>	void timer_register_update_event_select(uint32_t timer_periph, uint32_t reg, uint32_t event);
<b>Function descriptions</b>	select register reload 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)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>reg</b>	select shadow register
<i>TIMER_CH0CV_UPSEL</i>	select the TIMER_CH0CV register (TIMERx(x=0,7))
<i>TIMER_CH1CV_UPSEL</i>	select the TIMER_CH1CV register (TIMERx(x=0,7))
<i>TIMER_CH2CV_UPSEL</i>	select the TIMER_CH2CV register (TIMERx(x=0,7))
<i>TIMER_CH3CV_UPSEL</i>	select the TIMER_CH3CV register (TIMERx(x=0,7))
<i>TIMER_CH0CV_ADD_UPSEL</i>	select the TIMER_CH0MOVADD register (TIMERx(x=0,7))
<i>TIMER_CH1CV_ADD_UPSEL</i>	select the TIMER_CH1MOVADD register (TIMERx(x=0,7))

<i>TIMER_CH2CV_ADD_UPSEL</i>	select the TIMER_CH2MOVADD register (TIMERx(x=0,7))
<i>TIMER_CH3CV_ADD_UPSEL</i>	select the TIMER_CH3MOVADD register (TIMERx(x=0,7))
<i>TIMER_ADC_CR1_UPSEL</i>	select the TIMER_ADC_CCR1 register (TIMERx(x=0,7))
<i>TIMER_ADC_CR2_UPSEL</i>	select the TIMER_ADC_CCR2 register (TIMERx(x=0,7))
<i>TIMER_CAR_UPSEL</i>	select the TIMER_CAR register (TIMERx(x=0,7))
<i>TIMER_MCH0CV_UPSEL</i>	select the TIMER_MCH0CV register (TIMERx(x=0,7))
<i>TIMER_MCH1CV_UPSEL</i>	select the TIMER_MCH1CV register (TIMERx(x=0,7))
<i>TIMER_MCH2CV_UPSEL</i>	select the TIMER_MCH2CV register (TIMERx(x=0,7))
<i>TIMER_MCH3CV_UPSEL</i>	select the TIMER_MCH3CV register (TIMERx(x=0,7))
<b>Input parameter{in}</b>	
<b>event</b>	select update event
<i>TIMER_UPDATE_GLOBAL</i>	select global reload
<i>TIMER_UPDATE_NEXT_OVERFLOW</i>	select next overflow event
<i>TIMER_UPDATE_NEXT_UNDERFLOW</i>	select next underflow event
<i>TIMER_UPDATE_NEXT_FLOW</i>	select next flow event
<i>TIMER_UPDATE_NEXT_ADDCOMPARE_EVENT</i>	select next comapre event
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0 register reload event */
```

```
timer_register_update_event_select (TIMER0, TIMER_CH0CV_UPSEL, TIMER_UPDATE_GLOBAL);
```

### timer\_flag\_get

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

Table 3-959. 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
<i>TIMERx(x=0,1,2,7)</i>	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(x=0,1,2,7)</i>
<i>TIMER_FLAG_CH0</i>	TIMER channel 0 capture or compare flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_CH1</i>	TIMER channel 1 capture or compare flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_CH2</i>	TIMER channel 2 capture or compare flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_CH3</i>	TIMER channel 3 capture or compare flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_CMT</i>	TIMER commutation flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_TRG</i>	TIMER trigger flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_BRK</i>	TIMER break flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_CH0O</i>	TIMER channel 0 overcapture flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_CH1O</i>	TIMER channel 1 overcapture flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_CH2O</i>	TIMER channel 2 overcapture flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_CH3O</i>	TIMER channel 3 overcapture flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_FLAG_MCH0O</i>	TIMER multi mode channel 0 capture or compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH1O</i>	TIMER multi mode channel 1 capture or compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH2O</i>	TIMER multi mode channel 2 capture or compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH3O</i>	TIMER multi mode channel 3 capture or compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_SYSB</i>	TIMER System source break interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_UNDERFLOW</i>	TIMER cnt underflow interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_OVERFLOW</i>	TIMER cnt overflow interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH0O</i>	TIMER multi mode channel 0 overcapture flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH1O</i>	TIMER multi mode channel 1 overcapture flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH2O</i>	TIMER multi mode channel 2 overcapture flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH3O</i>	TIMER multi mode channel 3 overcapture flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_CH0CO MADD</i>	TIMER channel 0 additional compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_CH1CO MADD</i>	TIMER channel 1 additional compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_CH2CO MADD</i>	TIMER channel 2 additional compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_CH3CO</i>	TIMER channel 3 additional compare flag, <i>TIMERx(x=0,7)</i>

<i>MADD</i>	
<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 3-960. 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(x=0,1,2,7)</i>	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, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_CH0</i>	TIMER channel 0 capture or compare flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_CH1</i>	TIMER channel 1 capture or compare flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_CH2</i>	TIMER channel 2 capture or compare flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_CH3</i>	TIMER channel 3 capture or compare flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_CMT</i>	TIMER commutation flag, TIMERx(x=0,7)
<i>TIMER_FLAG_TRG</i>	TIMER trigger flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_BRK</i>	TIMER break flag, TIMERx(x=0,7)
<i>TIMER_FLAG_CH0O</i>	TIMER channel 0 overcapture flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_CH1O</i>	TIMER channel 1 overcapture flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_CH2O</i>	TIMER channel 2 overcapture flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_CH3O</i>	TIMER channel 3 overcapture flag, TIMERx(x=0,1,2,7)
<i>TIMER_FLAG_MCH0</i>	TIMER multi mode channel 0 capture or compare flag, TIMERx(x=0,7)
<i>TIMER_FLAG_MCH1</i>	TIMER multi mode channel 1 capture or compare flag, TIMERx(x=0,7)
<i>TIMER_FLAG_MCH2</i>	TIMER multi mode channel 2 capture or compare flag, TIMERx(x=0,7)
<i>TIMER_FLAG_MCH3</i>	TIMER multi mode channel 3 capture or compare flag, TIMERx(x=0,7)
<i>TIMER_FLAG_SYSB</i>	TIMER System source break interrupt flag, TIMERx(x=0,7)



<i>TIMER_FLAG_UNDERFLOW</i>	TIMER cnt underflow interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_OVERFLOW</i>	TIMER cnt overflow interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_MCH0O</i>	TIMER multi mode channel 0 overcapture flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_MCH1O</i>	TIMER multi mode channel 1 overcapture flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_MCH2O</i>	TIMER multi mode channel 2 overcapture flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_MCH3O</i>	TIMER multi mode channel 3 overcapture flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_CH0COMADD</i>	TIMER channel 0 additional compare flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_CH1COMADD</i>	TIMER channel 1 additional compare flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_CH2COMADD</i>	TIMER channel 2 additional compare flag, $TIMERx(x=0,7)$
<i>TIMER_FLAG_CH3COMADD</i>	TIMER channel 3 additional compare flag, $TIMERx(x=0,7)$
<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 3-961. 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
$TIMERx(x=0,1,2,7)$	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>interrupt</b>	timer interrupt enable source
<i>TIMER_INT_UP</i>	TIMER update interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH0</i>	TIMER channel 0 capture or compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH1</i>	TIMER channel 1 capture or compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH2</i>	TIMER channel 2 capture or compare interrupt, $TIMERx(x=0,1,2,7)$

<i>TIMER_INT_CH3</i>	TIMER channel 3 capture or compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CMT</i>	TIMER commutation interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_TRG</i>	TIMER trigger interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_BRK</i>	TIMER break interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_FLOW</i>	TIMER cnt flow interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_MCH0</i>	TIMER multi mode channel 0 capture or compare interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_MCH1</i>	TIMER multi mode channel 1 capture or compare interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_MCH2</i>	TIMER multi mode channel 2 capture or compare interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_MCH3</i>	TIMER multi mode channel 3 capture or compare interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_CH0COMA DD</i>	TIMER channel 0 additional compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH1COMA DD</i>	TIMER channel 1 additional compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH2COMA DD</i>	TIMER channel 2 additional compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH3COMA DD</i>	TIMER channel 3 additional compare interrupt, $TIMERx(x=0,1,2,7)$
<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 3-962. Function timer\_interrupt\_disable**

<b>Function name</b>	timer_interrupt_disable
<b>Function prototype</b>	void timer_interrupt_disable (uint32_t timer_periph, uint32_t interrupt);
<b>Function descriptions</b>	disable 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=0,1,2,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>interrupt</b>	timer interrupt source disable
<i>TIMER_INT_UP</i>	TIMER update interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH0</i>	TIMER channel 0 capture or compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH1</i>	TIMER channel 1 capture or compare interrupt, $TIMERx(x=0,1,2,7)$

<i>TIMER_INT_CH2</i>	TIMER channel 2 capture or compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH3</i>	TIMER channel 3 capture or compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CMT</i>	TIMER commutation interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_TRG</i>	TIMER trigger interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_BRK</i>	TIMER break interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_FLOW</i>	TIMER cnt flow interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_MCH0</i>	TIMER multi mode channel 0 capture or compare interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_MCH1</i>	TIMER multi mode channel 1 capture or compare interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_MCH2</i>	TIMER multi mode channel 2 capture or compare interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_MCH3</i>	TIMER multi mode channel 3 capture or compare interrupt, $TIMERx(x=0,7)$
<i>TIMER_INT_CH0COMA</i> <i>DD</i>	TIMER channel 0 additional compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH1COMA</i> <i>DD</i>	TIMER channel 1 additional compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH2COMA</i> <i>DD</i>	TIMER channel 2 additional compare interrupt, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_CH3COMA</i> <i>DD</i>	TIMER channel 3 additional compare interrupt, $TIMERx(x=0,1,2,7)$
<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 3-963. Function timer\_interrupt\_flag\_get**

<b>Function name</b>	timer_interrupt_flag_get
<b>Function prototype</b>	FlagStatus timer_interrupt_flag_get(uint32_t timer_periph, uint32_t int_flag);
<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(x=0,1,2,7)</i>	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>	TIMER update interrupt flag, $TIMERx(x=0,1,2,7)$

<i>TIMER_INT_FLAG_CH0</i>	TIMER channel 0 capture or compare interrupt flag, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_FLAG_CH1</i>	TIMER channel 1 capture or compare interrupt flag, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_FLAG_CH2</i>	TIMER channel 2 capture or compare interrupt flag, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_FLAG_CH3</i>	TIMER channel 3 capture or compare interrupt flag, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_FLAG_CMT</i>	TIMER channel commutation interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_TRG</i>	TIMER trigger interrupt flag, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_FLAG_BRK</i>	TIMER break interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_UNDERFLOW</i>	TIMER cnt underflow flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_OVERFLOW</i>	TIMER cnt overflow flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_MCH0</i>	TIMER multi mode channel 0 capture or compare interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_MCH1</i>	TIMER multi mode channel 1 capture or compare interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_MCH2</i>	TIMER multi mode channel 2 capture or compare interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_MCH3</i>	TIMER multi mode channel 3 capture or compare interrupt flag, $TIMERx(x=0,7)$
<i>TIMER_INT_FLAG_CH0COMADD</i>	TIMER channel 0 additional compare interrupt flag, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_FLAG_CH1COMADD</i>	TIMER channel 1 additional compare interrupt flag, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_FLAG_CH2COMADD</i>	TIMER channel 2 additional compare interrupt flag, $TIMERx(x=0,1,2,7)$
<i>TIMER_INT_FLAG_CH3COMADD</i>	TIMER channel 3 additional compare interrupt flag, $TIMERx(x=0,1,2,7)$
<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 3-964. 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
<i>TIMERx(x=0,1,2,7)</i>	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>	TIMER update interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CH0</i>	TIMER channel 0 capture or compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CH1</i>	TIMER channel 1 capture or compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CH2</i>	TIMER channel 2 capture or compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CH3</i>	TIMER channel 3 capture or compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CMT</i>	TIMER channel commutation interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_TRG</i>	TIMER trigger interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_BRK</i>	TIMER break interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_UNDERFLOW</i>	TIMER cnt underflow flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_OVERRFLOW</i>	TIMER cnt overflow flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_MCH0</i>	TIMER multi mode channel 0 capture or compare interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_MCH1</i>	TIMER multi mode channel 1 capture or compare interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_MCH2</i>	TIMER multi mode channel 2 capture or compare interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_MCH3</i>	TIMER multi mode channel 3 capture or compare interrupt flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_INT_FLAG_CH0COMADD</i>	TIMER channel 0 additional compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CH1COMADD</i>	TIMER channel 1 additional compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CH2COMADD</i>	TIMER channel 2 additional compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CH3COMADD</i>	TIMER channel 3 additional compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<b>int_flag</b>	the timer interrupt bits
<i>TIMER_INT_FLAG_UP</i>	TIMER update interrupt flag, <i>TIMERx(x=0,1,2,7)</i>
<i>TIMER_INT_FLAG_CH0</i>	TIMER channel 0 capture or compare interrupt flag, <i>TIMERx(x=0,1,2,7)</i>

<i>TIMER_INT_FLAG_CH1</i>	TIMER channel 1 capture or compare interrupt flag, TIMERx(x=0,1,2,7)
<i>TIMER_INT_FLAG_CH2</i>	TIMER channel 2 capture or compare interrupt flag, TIMERx(x=0,1,2,7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear TIMER0 update interrupt flag */
```

```
timer_interrupt_flag_clear(TIMER0, TIMER_INT_FLAG_UP);
```

## 3.28. 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 [3.28.1](#), the TMU firmware functions are introduced in chapter [3.28.2](#).

### 3.28.1. Descriptions of Peripheral registers

TMU registers are listed in the table shown as below:

**Table 3-965. TMU Registers**

Registers	Descriptions
TMU_CS	TMU control and status register
TMU_IDATA	TMU input data register
TMU_ODATA	TMU output data register

### 3.28.2. Descriptions of Peripheral functions

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

**Table 3-966. 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
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

Function name	Function description
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 3-967. 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_SQRT)
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, 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 3-968. Function tmu\_deinit**

Function name	tmu_deinit
Function prototype	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 3-969. 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>	
<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 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 3-970. Function tmu\_init**

<b>Function name</b>	tmu_init
<b>Function prototype</b>	void tmu_init(tmu_parameter_struct* init_struct);



Function descriptions	initialize TMU
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	pointer to TMU init parameter struct, the structure members can refer to <a href="#">Structure tmu_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize TMU */

tmu_parameter_struct tmu_init_struct;

tmu_init_struct.mode = TMU_MODE_COS;

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 3-971. Function tmu\_dma\_read\_enable**

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

Output parameter{out}	
-	-
Return value	
-	-

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 3-972. Function tmu\_dma\_read\_disable**

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

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 3-973. Function tmu\_dma\_write\_enable**

Function name	tmu_dma_write_enable
Function prototype	void tmu_dma_write_enable(void);
Function descriptions	enable TMU write interrupt
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-

Return value	
-	-

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 3-974. 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 3-975. 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 3-976. Function tmu\_two\_q31\_write**

Function name	tmu_two_q31_write
Function prototype	void tmu_two_q31_write(uint32_t data1, uint32_t data2);
Function descriptions	write two data in q1.31 format
Precondition	-
The called functions	-
Input parameter{in}	
data1	the first input data in q1.31 format (0x00000000~0xFFFFFFFF)
Input parameter{in}	
data2	the second input data in q1.31 format (0x00000000~0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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 3-977. Function tmu\_two\_q15\_write**

Function name	tmu_two_q15_write
Function prototype	void tmu_two_q15_write(uint16_t data1, uint16_t data2);
Function descriptions	write two data in q1.15 format
Precondition	-
The called functions	-
Input parameter{in}	
data1	the first input data in q1.15 format (0x0000~0xFFFF)

Input parameter{in}	
<b>data2</b>	the second input data in q1.15 format (0x0000~0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

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 3-978. 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>	-
Input parameter{in}	
<b>data</b>	the input data in floating point format
Output parameter{out}	
-	-
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 3-979. 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 3-980. 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 3-981. Function tmu\_two\_q31\_read

Function name	tmu_two_q31_read
Function prototype	void tmu_two_q31_read(uint32_t* p1, uint32_t* p2);
Function descriptions	read two data in q1.31 format
Precondition	-
The called functions	-
Input parameter{in}	
p1	A pointer to the first output data(q1.31 format)
Input parameter{in}	
p2	A pointer to the second output data(q1.31 format)
Output parameter{out}	
-	-
Return value	
-	-

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 3-982. 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 3-983. Function tmu\_one\_f32\_read**

<b>Function name</b>	tmu_one_f32_read
<b>Function prototype</b>	void tmu_one_f32_read(float* p);
<b>Function descriptions</b>	read one data in floating point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>p</b>	A pointer to output data(floating point format)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-984. Function tmu\_two\_f32\_read**

<b>Function name</b>	tmu_two_f32_read
<b>Function prototype</b>	void tmu_two_f32_read(float* p1, float* p2)
<b>Function descriptions</b>	read two data in floating point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>p1</b>	A pointer to the first output data(floating point format)
<b>Input parameter{in}</b>	
<b>p2</b>	A pointer to the second output data(floating point format)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	



-	-
---	---

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 3-985. Function tmu\_flag\_get**

<b>Function name</b>	tmu_flag_get
<b>Function prototype</b>	FlagStatus tmu_flag_get(uint32_t flag);
<b>Function descriptions</b>	get TMU flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-986. Function tmu\_flag\_clear**

<b>Function name</b>	tmu_flag_clear
<b>Function prototype</b>	void tmu_flag_clear(uint32_t flag);
<b>Function descriptions</b>	clear TMU flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>flag</b>	the TMU flags
<i>TMU_FLAG_OVRF</i>	TMU overflow error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-987. Function tmu\_interrupt\_enable**

<b>Function name</b>	tmu_interrupt_enable
<b>Function prototype</b>	void tmu_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable 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:

```
/* enable TMU interrupt */
tmu_interrupt_enable(TMU_INT_OVRF);
```

### tmu\_interrupt\_disable

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

**Table 3-988. 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>	-

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

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 3-989. 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_OVRF</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 3-990. 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>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	the TMU interrupt flags
<i>TMU_INT_FLAG_OVR F</i>	TMU overflow interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
tmu_interrupt_flag_clear(TMU_INT_FLAG_OVRF);
```

## 3.29. UART

The Universal Asynchronous Receiver/Transmitter (UART) provides a flexible serial data exchange interface. The UART registers are listed in chapter [3.29.1](#), the UART firmware functions are introduced in chapter [3.29.2](#).

### 3.29.1. Descriptions of Peripheral registers

UART registers are listed in the table shown as below:

**Table 3-991. UART Registers**

Registers	Descriptions
UART_STAT0	Status register 0
UART_DATA	Data register
UART_BAUD	Baud rate register
UART_CTL0	Control register 0
UART_CTL1	Control register 1
UART_CTL2	Control register 2
UART_GP	Guard time and prescaler register
UART_CHC	Coherence control register

### 3.29.2. Descriptions of Peripheral functions

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

Table 3-992. UART firmware function

Function name	Function description
uart_deinit	reset UART
uart_baudrate_set	configure UART baud rate value
uart_parity_config	configure UART parity function
uart_word_length_set	configure UART word length
uart_stop_bit_set	configure UART stop bit length
uart_enable	enable UART
uart_disable	disable UART
uart_transmit_config	configure UART transmitter
uart_receive_config	configure UART receiver
uart_data_first_config	data is transmitted/received with the LSB/MSB first
uart_invert_config	configure UART inverted
uart_oversample_config	configure the UART oversample mode
uart_sample_bit_config	configure sample bit method
uart_data_transmit	UART transmit data function
uart_data_receive	UART receive data function
uart_address_config	configure address of the UART
uart_mute_mode_enable	enable mute mode
uart_mute_mode_disable	disable mute mode
uart_mute_mode_wakeup_config	configure wakeup method in mute mode
uart_lin_mode_enable	enable LIN mode
uart_lin_mode_disable	disable LIN mode
uart_lin_break_detection_length_config	LIN break detection length
uart_send_break	send break frame
uart_halfduplex_enable	enable half-duplex mode
uart_halfduplex_disable	disable half-duplex mode
uart_irda_mode_enable	enable IrDA mode
uart_irda_mode_disable	disable IrDA mode
uart_prescaler_config	configure the peripheral clock prescaler
uart_irda_lowpower_config	configure IrDA low-power
uart_parity_check_coherence_config	configure parity check coherence mode
uart_dma_receive_config	configure UART DMA for reception
uart_dma_transmit_config	configure UART DMA for transmission
uart_flag_get	get flag in STAT/RFCFS register
uart_flag_clear	clear UART status
uart_interrupt_enable	enable UART interrupt
uart_interrupt_disable	disable UART interrupt
uart_interrupt_flag_get	get UART interrupt and flag status
uart_interrupt_flag_clear	clear UART interrupt flag

## Enum uart\_flag\_enum

**Table 3-993. Enum uart\_flag\_enum**

Member name	Function description
UART_FLAG_LBD	LIN break detected flag
UART_FLAG_TBE	transmit data buffer empty
UART_FLAG_TC	transmission complete
UART_FLAG_RBNE	read data buffer not empty
UART_FLAG_IDLE	IDLE line detected flag
UART_FLAG_ORERR	overrun error
UART_FLAG_NERR	noise error flag
UART_FLAG_FERR	frame error flag
UART_FLAG_PERR	parity error flag
UART_FLAG_EPERR	early parity error flag

## Enum uart\_interrupt\_flag\_enum

**Table 3-994. Enum uart\_interrupt\_flag\_enum**

Member name	Function description
UART_INT_FLAG_PERR	parity error interrupt and flag
UART_INT_FLAG_TBE	transmitter buffer empty interrupt and flag
UART_INT_FLAG_TC	transmission complete interrupt and flag
UART_INT_FLAG_RBNE	read data buffer not empty interrupt and flag
UART_INT_FLAG_RBNE_ORERR	read data buffer not empty interrupt and overrun error flag
UART_INT_FLAG_IDLE	IDLE line detected interrupt and flag
UART_INT_FLAG_LBD	LIN break detected interrupt and flag
UART_INT_FLAG_ERR_ORERR	error interrupt and overrun error
UART_INT_FLAG_ERR_NERR	error interrupt and noise error flag
UART_INT_FLAG_ERR_FERR	error interrupt and frame error flag

## Enum uart\_interrupt\_enum

**Table 3-995. Enum uart\_interrupt\_enum**

Member name	Function description
UART_INT_PERR	parity error interrupt
UART_INT_TBE	transmitter buffer empty interrupt
UART_INT_TC	transmission complete interrupt
UART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt
UART_INT_IDLE	IDLE line detected interrupt
UART_INT_LBD	LIN break detected interrupt
UART_INT_ERR	error interrupt

## Enum uart\_invert\_enum

Table 3-996. Enum uart\_invert\_enum

Member name	Function description
UART_DINV_ENABLE	data bit level inversion
UART_DINV_DISABLE	data bit level not inversion
UART_TXPIN_ENABLE	TX pin level inversion
UART_TXPIN_DISABLE	TX pin level not inversion
UART_RXPIN_ENABLE	RX pin level inversion
UART_RXPIN_DISABLE	RX pin level not inversion

## uart\_deinit

The description of uart\_deinit is shown as below:

Table 3-997. Function uart\_deinit

Function name	uart_deinit
Function prototype	void uart_deinit(uint32_t uart_periph);
Function descriptions	reset UART
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
uart_periph	uart peripheral
UARTx	x=0,1,2,3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset UART0 */
uart_deinit(UART0);
```

## uart\_baudrate\_set

The description of uart\_baudrate\_set is shown as below:

Table 3-998. Function uart\_baudrate\_set

Function name	uart_baudrate_set
Function prototype	void uart_baudrate_set(uint32_t uart_periph, uint32_t baudval);
Function descriptions	configure UART baud rate value
Precondition	-
The called functions	rcu_clock_freq_get
Input parameter{in}	
uart_periph	uart peripheral

<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>baudval</b>	baud rate value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure UART0 baud rate value */
```

```
uart_baudrate_set(UART0, 115200);
```

### uart\_parity\_config

The description of uart\_parity\_config is shown as below:

**Table 3-999. Function uart\_parity\_config**

<b>Function name</b>	uart_parity_config
<b>Function prototype</b>	void uart_parity_config(uint32_t uart_periph, uint32_t paritycfg);
<b>Function descriptions</b>	configure UART parity function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>paritycfg</b>	configure UART parity
<i>UART_PM_NONE</i>	no parity
<i>UART_PM_ODD</i>	odd parity
<i>UART_PM_EVEN</i>	even parity
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure UART0 parity */
```

```
uart_parity_config(UART0, UART_PM_EVEN);
```

### uart\_word\_length\_set

The description of uart\_word\_length\_set is shown as below:



Table 3-1000. Function `uart_word_length_set`

<b>Function name</b>	<code>uart_word_length_set</code>
<b>Function prototype</b>	<code>void uart_word_length_set(uint32_t uart_periph, uint32_t wlen);</code>
<b>Function descriptions</b>	configure UART word length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>wlen</b>	UART word length configure
<i>UART_WL_8BIT</i>	8 bits
<i>UART_WL_9BIT</i>	9 bits
<i>UART_WL_7BIT</i>	7 bits
<i>UART_WL_10BIT</i>	10 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure UART0 word length */
```

```
uart_word_length_set(UART0, UART_WL_9BIT);
```

### `uart_stop_bit_set`

The description of `uart_stop_bit_set` is shown as below:

Table 3-1001. Function `uart_stop_bit_set`

<b>Function name</b>	<code>uart_stop_bit_set</code>
<b>Function prototype</b>	<code>void uart_stop_bit_set(uint32_t uart_periph, uint32_t stblen);</code>
<b>Function descriptions</b>	configure UART stop bit length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>stblen</b>	UART stop bit configure
<i>UART_STB_1BIT</i>	1 bit
<i>UART_STB_2BIT</i>	2 bits
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* configure UART0 stop bit length */
uart_stop_bit_set(UART0, UART_STB_1BIT);
```

### uart\_enable

The description of uart\_enable is shown as below:

**Table 3-1002. Function uart\_enable**

Function name	uart_enable
Function prototype	void uart_enable(uint32_t uart_periph);
Function descriptions	enable UART
Precondition	-
The called functions	-
Input parameter{in}	
uart_periph	uart peripheral
UARTx	x=0,1,2,3
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable UART0 */
uart_enable(UART0);
```

### uart\_disable

The description of uart\_disable is shown as below:

**Table 3-1003. Function uart\_disable**

Function name	uart_disable
Function prototype	void uart_disable(uint32_t uart_periph);
Function descriptions	disable UART
Precondition	-
The called functions	-
Input parameter{in}	
uart_periph	uart peripheral
UARTx	x=0,1,2,3
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* disable UART0 */
```

```
uart_disable(UART0);
```

### uart\_transmit\_config

The description of uart\_transmit\_config is shown as below:

**Table 3-1004. Function uart\_transmit\_config**

Function name	uart_transmit_config
Function prototype	void uart_transmit_config(uint32_t uart_periph, uint32_t txconfig);
Function descriptions	configure UART transmitter
Precondition	-
The called functions	-
Input parameter{in}	
uart_periph	uart peripheral
UARTx	x=0,1,2,3
Input parameter{in}	
txconfig	enable or disable UART transmitter
UART_TRANSMIT_ENABLE	enable UART transmission
UART_TRANSMIT_DISABLE	disable UART transmission
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure UART0 transmitter */
```

```
uart_transmit_config(UART0, UART_TRANSMIT_ENABLE);
```

### uart\_receive\_config

The description of uart\_receive\_config is shown as below:

**Table 3-1005. Function uart\_receive\_config**

Function name	uart_receive_config
Function prototype	void uart_receive_config(uint32_t uart_periph, uint32_t rxconfig);
Function descriptions	configure UART receiver
Precondition	-

The called functions	-
Input parameter{in}	
uart_periph	uart peripheral
UARTx	x=0,1,2,3
Input parameter{in}	
rxconfig	enable or disable UART receiver
UART_RECEIVE_ENABLE	enable UART reception
UART_RECEIVE_DISABLE	disable UART reception
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure UART0 receiver */
```

```
uart_receive_config(UART0, UART_RECEIVE_ENABLE);
```

### uart\_data\_first\_config

The description of uart\_data\_first\_config is shown as below:

**Table 3-1006. Function uart\_data\_first\_config**

Function name	uart_data_first_config
Function prototype	void uart_data_first_config(uint32_t uart_periph, uint32_t msbf);
Function descriptions	data is transmitted/received with the LSB/MSB first
Precondition	-
The called functions	-
Input parameter{in}	
uart_periph	uart peripheral
UARTx	x=0,1,2,3
Input parameter{in}	
msbf	LSB/MSB
UART_MSBF_LSB	LSB first
UART_MSBF_MSB	MSB first
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure LSB of data first */
```

```
uart_data_first_config(UART0, UART_MSBF_LSB);
```

## uart\_invert\_config

The description of `uart_invert_config` is shown as below:

**Table 3-1007. Function `uart_invert_config`**

<b>Function name</b>	<code>uart_invert_config</code>
<b>Function prototype</b>	<code>void uart_invert_config(uint32_t uart_periph, uart_invert_enum invertpara);</code>
<b>Function descriptions</b>	configure UART inverted
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>invertpara</b>	refer to <a href="#">Table 3-996. Enum <code>uart_invert_enum</code></a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure UART0 inversion */
```

```
uart_invert_config(UART0, UART_DINV_ENABLE);
```

## uart\_oversample\_config

The description of `uart_oversample_config` is shown as below:

**Table 3-1008. Function `uart_oversample_config`**

<b>Function name</b>	<code>uart_oversample_config</code>
<b>Function prototype</b>	<code>void uart_oversample_config(uint32_t uart_periph, uint32_t oversamp);</code>
<b>Function descriptions</b>	configure the UART oversample mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>oversamp</b>	oversample value
<i>UART_OVSMOD_8</i>	oversampling by 8
<i>UART_OVSMOD_16</i>	oversampling by 16
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* config UART0 oversampling by 8 */
```

```
uart_oversample_config(UART0, UART_OVSMOD_8);
```

### uart\_sample\_bit\_config

The description of uart\_sample\_bit\_config is shown as below:

**Table 3-1009. Function uart\_sample\_bit\_config**

Function name	uart_sample_bit_config
Function prototype	void uart_sample_bit_config(uint32_t uart_periph, uint32_t osb);
Function descriptions	configure the sample bit method
Precondition	-
The called functions	-
Input parameter{in}	
uart_periph	uart peripheral
UARTx	x=0,1,2,3
Input parameter{in}	
osb	sample bit
UART_OSB_1BIT	1 bit
UART_OSB_3BIT	3 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config UART0 1 bit sample mode */
```

```
uart_sample_bit_config(UART0, UART_OSB_1BIT);
```

### uart\_data\_transmit

The description of uart\_data\_transmit is shown as below:

**Table 3-1010. Function uart\_data\_transmit**

Function name	uart_data_transmit
Function prototype	void uart_data_transmit(uint32_t uart_periph, uint16_t data);
Function descriptions	UART transmit data function
Precondition	-
The called functions	-
Input parameter{in}	

<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>data</b>	data of transmission (0x00-0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* UART0 transmit data */
```

```
uart_data_transmit(UART0, 0xAA);
```

### uart\_data\_receive

The description of uart\_data\_receive is shown as below:

**Table 3-1011. Function uart\_data\_receive**

<b>Function name</b>	uart_data_receive
<b>Function prototype</b>	uint16_t uart_data_receive(uint32_t uart_periph);
<b>Function descriptions</b>	UART receive data function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	data of received (0x00-0xFF)

Example:

```
/* UART0 receive data */
```

```
uint16_t temp;
```

```
temp = uart_data_receive(UART0);
```

### uart\_address\_config

The description of uart\_address\_config is shown as below:

**Table 3-1012. Function uart\_address\_config**

<b>Function name</b>	uart_address_config
<b>Function prototype</b>	void uart_address_config(uint32_t uart_periph, uint8_t addr);
<b>Function descriptions</b>	configure the address of the UART

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>addr</b>	address of UART (0x00-0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure address of the UART0 */
```

```
uart_address_config(UART0, 0x00);
```

### uart\_mute\_mode\_enable

The description of uart\_mute\_mode\_enable is shown as below:

**Table 3-1013. Function uart\_mute\_mode\_enable**

<b>Function name</b>	uart_mute_mode_enable
<b>Function prototype</b>	void uart_mute_mode_enable(uint32_t uart_periph);
<b>Function descriptions</b>	enable mute mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable UART0 receiver in mute mode */
```

```
uart_mute_mode_enable(UART0);
```

### uart\_mute\_mode\_disable

The description of uart\_mute\_mode\_disable is shown as below:

**Table 3-1014. Function uart\_mute\_mode\_disable**

<b>Function name</b>	uart_mute_mode_disable
----------------------	------------------------



<b>Function prototype</b>	void uart_mute_mode_disable(uint32_t uart_periph);
<b>Function descriptions</b>	disable mute mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable UART0 receiver in mute mode */
```

```
uart_mute_mode_disable(UART0);
```

### uart\_mute\_mode\_wakeup\_config

The description of uart\_mute\_mode\_wakeup\_config is shown as below:

**Table 3-1015. Function uart\_mute\_mode\_wakeup\_config**

<b>Function name</b>	uart_mute_mode_wakeup_config
<b>Function prototype</b>	void uart_mute_mode_wakeup_config(uint32_t uart_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>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>wmethod</b>	two methods be used to enter or exit the mute mode
<i>UART_WM_IDLE</i>	idle line
<i>UART_WM_ADDR</i>	address mask
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure UART0 wakeup method in mute mode */
```

```
uart_mute_mode_wakeup_config(UART0, UART_WM_IDLE);
```

## uart\_lin\_mode\_enable

The description of uart\_lin\_mode\_enable is shown as below:

**Table 3-1016. Function uart\_lin\_mode\_enable**

<b>Function name</b>	uart_lin_mode_enable
<b>Function prototype</b>	void uart_lin_mode_enable(uint32_t uart_periph);
<b>Function descriptions</b>	enable LIN mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* UART0 LIN mode enable */
uart_lin_mode_enable(UART0);
```

## uart\_lin\_mode\_disable

The description of uart\_lin\_mode\_disable is shown as below:

**Table 3-1017. Function uart\_lin\_mode\_disable**

<b>Function name</b>	uart_lin_mode_disable
<b>Function prototype</b>	void uart_lin_mode_disable(uint32_t uart_periph);
<b>Function descriptions</b>	disable LIN mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* UART0 LIN mode disable */
uart_lin_mode_disable(UART0);
```

## uart\_lin\_break\_dection\_length\_config

The description of uart\_lin\_break\_dection\_length\_config is shown as below:

**Table 3-1018. Function uart\_lin\_break\_dection\_length\_config**

<b>Function name</b>	uart_lin_break_dection_length_config
<b>Function prototype</b>	void uart_lin_break_dection_length_config(uint32_t uart_periph, uint32_t lblen);
<b>Function descriptions</b>	LIN break detection length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>lblen</b>	two methods be used to enter or exit the mute mode
<i>UART_LBLEN_10B</i>	10 bits
<i>UART_LBLEN_11B</i>	11 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure LIN break frame length */
```

```
uart_lin_break_dection_length_config(UART0, UART_LBLEN_10B);
```

## uart\_send\_break

The description of uart\_send\_break is shown as below:

**Table 3-1019. Function uart\_halfduplex\_enable**

<b>Function name</b>	uart_send_break
<b>Function prototype</b>	void uart_send_break(uint32_t uart_periph);
<b>Function descriptions</b>	send break frame
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* send break frame */

uart_send_break(UART0);
```

### uart\_halfduplex\_enable

The description of uart\_halfduplex\_enable is shown as below:

**Table 3-1020. Function uart\_halfduplex\_enable**

<b>Function name</b>	uart_halfduplex_enable
<b>Function prototype</b>	void uart_halfduplex_enable(uint32_t uart_periph);
<b>Function descriptions</b>	enable half-duplex mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable UART0 half duplex mode*/

uart_halfduplex_enable(UART0);
```

### uart\_halfduplex\_disable

The description of uart\_halfduplex\_disable is shown as below:

**Table 3-1021. Function uart\_halfduplex\_disable**

<b>Function name</b>	uart_halfduplex_disable
<b>Function prototype</b>	void uart_halfduplex_disable(uint32_t uart_periph);
<b>Function descriptions</b>	disable half-duplex mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable UART0 half duplex mode*/

uart_halfduplex_disable(UART0);
```

### uart\_irda\_mode\_enable

The description of uart\_irda\_mode\_enable is shown as below:

**Table 3-1022. Function uart\_irda\_mode\_enable**

<b>Function name</b>	uart_irda_mode_enable
<b>Function prototype</b>	void uart_irda_mode_enable(uint32_t uart_periph);
<b>Function descriptions</b>	enable IrDA mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable UART0 IrDA mode */

uart_irda_mode_enable(UART0);
```

### uart\_irda\_mode\_disable

The description of uart\_irda\_mode\_disable is shown as below:

**Table 3-1023. Function uart\_irda\_mode\_disable**

<b>Function name</b>	uart_irda_mode_disable
<b>Function prototype</b>	void uart_irda_mode_disable(uint32_t uart_periph);
<b>Function descriptions</b>	disable IrDA mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable UART0 IrDA mode */
uart_irda_mode_disable(UART0);
```

### uart\_prescaler\_config

The description of uart\_prescaler\_config is shown as below:

**Table 3-1024. Function uart\_prescaler\_config**

<b>Function name</b>	uart_prescaler_config
<b>Function prototype</b>	void uart_prescaler_config(uint32_t uart_periph, uint8_t psc);
<b>Function descriptions</b>	configure the peripheral clock prescaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>psc</b>	clock prescaler (0x00-0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the UART0 peripheral clock prescaler in UART IrDA low-power mode */
uart_prescaler_config(UART0, 0x00);
```

### uart\_irda\_lowpower\_config

The description of uart\_irda\_lowpower\_config is shown as below:

**Table 3-1025. Function uart\_irda\_lowpower\_config**

<b>Function name</b>	uart_irda_lowpower_config
<b>Function prototype</b>	void uart_irda_lowpower_config(uint32_t uart_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>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>irlp</b>	IrDA low-power or normal
<i>UART_IRLP_LOW</i>	low-power

<i>UART_IRLP_NORMAL</i>	normal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure UART0 IrDA low-power */
uart_irda_lowpower_config(UART0, UART_IRLP_LOW);
```

### uart\_parity\_check\_coherence\_config

The description of `uart_parity_check_coherence_config` is shown as below:

**Table 3-1026. Function `uart_parity_check_coherence_config`**

<b>Function name</b>	<code>uart_parity_check_coherence_config</code>
<b>Function prototype</b>	<code>void uart_parity_check_coherence_config(uint32_t uart_periph, uint32_t pcm);</code>
<b>Function descriptions</b>	configure parity check coherence mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<i>pcm</i>	
<i>UART_PCM_NONE</i>	not check parity
<i>UART_PCM_EN</i>	check the parity
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enabled UART0 configure parity check coherence */
uart_parity_check_coherence_config (UART0, UART_PCM_EN);
```

### uart\_dma\_receive\_config

The description of `uart_dma_receive_config` is shown as below:

**Table 3-1027. Function `uart_dma_receive_config`**

<b>Function name</b>	<code>uart_dma_receive_config</code>
<b>Function prototype</b>	<code>void uart_dma_receive_config(uint32_t uart_periph, uint32_t dmacmd);</code>

<b>Function descriptions</b>	configure UART DMA for reception
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>dmacmd</b>	enable or disable DMA for reception
<i>UART_RECEIVE_DMA_ENABLE</i>	DMA enable for reception
<i>UART_RECEIVE_DMA_DISABLE</i>	DMA disable for reception
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* UART0 DMA enable for reception */
```

```
uart_dma_receive_config(UART0, UART_RECEIVE_DMA_ENABLE);
```

### uart\_dma\_transmit\_config

The description of uart\_dma\_transmit\_config is shown as below:

**Table 3-1028. Function uart\_dma\_transmit\_config**

<b>Function name</b>	uart_dma_transmit_config
<b>Function prototype</b>	void uart_dma_transmit_config(uint32_t uart_periph, uint32_t dmacmd);
<b>Function descriptions</b>	configure UART DMA for transmission
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>dmacmd</b>	enable or disable DMA for transmission
<i>UART_TRANSMIT_DMA_ENABLE</i>	DMA enable for transmission
<i>UART_TRANSMIT_DMA_DISABLE</i>	DMA disable for transmission
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-



Example:

```
/* UART0 DMA enable for transmission */
```

```
uart_dma_transmit_config(UART0, UART_TRANSMIT_DMA_ENABLE);
```

### uart\_flag\_get

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

**Table 3-1029. Function uart\_flag\_get**

<b>Function name</b>	uart_flag_get
<b>Function prototype</b>	FlagStatus uart_flag_get(uint32_t uart_periph, uart_flag_enum flag);
<b>Function descriptions</b>	get flag in STAT/RFCs register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>flag</b>	UART flags, refer to <a href="#">Table 3-993. Enum uart_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 UART0 state */
```

```
FlagStatus status;
```

```
status = uart_flag_get(UART0, UART_FLAG_TBE);
```

### uart\_flag\_clear

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

**Table 3-1030. Function uart\_flag\_clear**

<b>Function name</b>	uart_flag_clear
<b>Function prototype</b>	void uart_flag_clear(uint32_t uart_periph, uart_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>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3

Input parameter{in}	
flag	UART flags, refer to <a href="#">Table 3-993. Enum uart_flag_enum</a> only one among these parameters can be selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear UART0 flag */
uart_flag_clear(UART0, UART_FLAG_TC);
```

### uart\_interrupt\_enable

The description of `uart_interrupt_enable` is shown as below:

**Table 3-1031. Function `uart_interrupt_enable`**

Function name	uart_interrupt_enable
Function prototype	void uart_interrupt_enable(uint32_t uart_periph, uart_interrupt_enum interrupt);
Function descriptions	enable UART interrupt
Precondition	-
The called functions	-
Input parameter{in}	
uart_periph	uart peripheral
UARTx	x=0,1,2,3
Input parameter{in}	
interrupt	interrupt type, refer to <a href="#">Table 3-995. Enum uart_interrupt_enum</a> only one among these parameters can be selected
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable UART0 TBE interrupt */
uart_interrupt_enable(UART0, UART_INT_TBE);
```

### uart\_interrupt\_disable

The description of `uart_interrupt_disable` is shown as below:

**Table 3-1032. Function `uart_interrupt_disable`**

Function name	uart_interrupt_disable
---------------	------------------------

<b>Function prototype</b>	void uart_interrupt_disable(uint32_t uart_periph, uart_interrupt_enum interrupt);
<b>Function descriptions</b>	disable UART interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>interrupt</b>	interrupt type, refer to <a href="#">Table 3-995. Enum uart_interrupt_enum</a> only one among these parameters can be selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable UART0 TBE interrupt */
```

```
uart_interrupt_disable(UART0, UART_INT_TBE);
```

### uart\_interrupt\_flag\_get

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

**Table 3-1033. Function uart\_interrupt\_flag\_get**

<b>Function name</b>	uart_interrupt_flag_get
<b>Function prototype</b>	FlagStatus uart_interrupt_flag_get(uint32_t uart_periph, uart_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	get UART interrupt and flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>int_flag</b>	UART interrupt flag, refer to <a href="#">Table 3-994. Enum uart_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 UART0 interrupt flag status */
```

```
FlagStatus status;
```

```
status = uart_interrupt_flag_get(UART0, UART_INT_FLAG_RBNE);
```

### uart\_interrupt\_flag\_clear

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

**Table 3-1034. Function uart\_interrupt\_flag\_clear**

<b>Function name</b>	uart_interrupt_flag_clear
<b>Function prototype</b>	void uart_interrupt_flag_clear(uint32_t uart_periph, uart_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	clear UART interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>uart_periph</b>	uart peripheral
<i>UARTx</i>	x=0,1,2,3
<b>Input parameter{in}</b>	
<b>int_flag</b>	UART interrupt flag, refer to <a href="#">Table 3-994. Enum uart_interrupt_flag_enum</a> , only one among these parameters can be selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the UART0 interrupt flag */
```

```
uart_interrupt_flag_clear(UART0, UART_INT_FLAG_TC);
```

## 3.30. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. The WWDGT registers are listed in chapter [3.30.1](#), the WWDGT firmware functions are introduced in chapter [3.30.2](#).

### 3.30.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

**Table 3-1035. WWDGT Registers**

Registers	Descriptions
WWDGT_CTL	WWDGT control register
WWDGT_CFG	WWDGT configuration register
WWDGT_STAT	WWDGT status register

### 3.30.2. Descriptions of Peripheral functions

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

**Table 3-1036. WWDGT firmware function**

Function name	Function description
wwdgt_deinit	reset the window watchdog timer configuration
wwdgt_counter_update	configure the WWDGT counter value
wwdgt_struct_para_init	initialize the parameters of WWDGT configure structure with the default values
wwdgt_cfg_init	initialize WWDGT configuration and start counter
wwdgt_flag_get	check flag state of WWDGT
wwdgt_flag_clear	check flag state of WWDGT

#### Structure wwdgt\_cfg\_parameter\_struct

**Table 3-1037. Structure wwdgt\_cfg\_parameter\_struct**

Member name	Function description
counter	watchdog timer counter value
reset_control	WWDGT reset control bit
ewie_control	WWDGT early wakeup interrupt control bit
window	WWDGT window start value
window_end_position	WWDGT window end position value
prescaler	WWDGT prescaler value

#### wwdgt\_deinit

The description of wwdgt\_deinit is shown as below:

**Table 3-1038. 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\_counter\_update

The description of wwdgt\_counter\_update is shown as below:

**Table 3-1039. Function wwdgt\_counter\_update**

Function name	wwdgt_counter_update
Function prototype	void wwdgt_counter_update(uint16_t counter_value);
Function descriptions	configure the window watchdog timer counter value
Precondition	-
The called functions	-
Input parameter{in}	
counter_value	counter_value: 0x0000 - 0x3FFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* update WWDGT counter to 0x3FFF */
```

```
wwdgt_counter_update(0x3FFF);
```

### wwdgt\_struct\_para\_init

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

**Table 3-1040. Function wwdgt\_struct\_para\_init**

Function name	wwdgt_struct_para_init
Function prototype	void wwdgt_struct_para_init(wwdgt_cfg_parameter_struct *cfg_struct);
Function descriptions	initialize the parameters of WWDGT configure structure with the default values
Precondition	-
The called functions	-
Input parameter{in}	
cfg_struct	Structure for initialization, the structure members can refer to <a href="#">Table 3-1037</a> . <a href="#">Structure wwdgt_cfg_parameter_struct</a>
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* initialize the parameters of WWDGT */
wwdgt_cfg_parameter_struct wwdgt_cfg_struct;
wwdgt_struct_para_init(&wwdgt_cfg_struct);
```

## wwdgt\_cfg\_init

The description of wwdgt\_cfg\_init is shown as below:

**Table 3-1041. Function wwdgt\_cfg\_init**

Function name	wwdgt_cfg_init
Function prototype	void wwdgt_cfg_init(wwdgt_cfg_parameter_struct *cfg_struct);
Function descriptions	initialize WWDGT configuration and start counter
Precondition	-
The called functions	-
Input parameter{in}	
cfg_struct	Structure for initialization, the structure members can refer to <a href="#">Table 3-1037. Structure wwdgt_cfg_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize WWDGT */
wwdgt_cfg_parameter_struct wwdgt_cfg_struct;
wwdgt_struct_para_init(&wwdgt_cfg_struct);

wwdgt_cfg_struct.counter = 16383U;
wwdgt_cfg_struct.reset_control = WWDGT_RESET_ENABLE;
wwdgt_cfg_struct.ewie_control = WWDGT_EWIE_DISABLE;
wwdgt_cfg_struct.window = 12000U;
wwdgt_cfg_struct.window_end_position = WWDGT_CFG_WEPS_THREE_QUARTER;
wwdgt_cfg_struct.prescaler = WWDGT_CFG_PSC_DIV8;
wwdgt_cfg_init(&wwdgt_cfg_struct);
```

## wwdgt\_interrupt\_enable

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

**Table 3-1042. Function wwdgt\_interrupt\_enable**

Function name	wwdgt_interrupt_enable
---------------	------------------------

<b>Function prototype</b>	void wwdgt_interrupt_enable(void);
<b>Function descriptions</b>	enable early wakeup interrupt 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:

```
/* enable early wakeup interrupt of WWDGT */
```

```
wwdgt_interrupt_enable();
```

### wwdgt\_flag\_get

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

**Table 3-1043. Function wwdgt\_flag\_get**

<b>Function name</b>	wwdgt_flag_get
<b>Function prototype</b>	FlagStatus wwdgt_flag_get(uint32_t flag);
<b>Function descriptions</b>	check flag state of WWDGT
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	flag state of WWDGT
WWDGT_FLAG_EWIF	early wakeup interrupt flag
WWDGT_FLAG_REFE F	refresh error flag
WWDGT_FLAG_UNDF F	underflow error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* test if early wakeup interrupt flag is set */
```

```
FlagStatus status;
```

```
status = wwdgt_flag_get(WWDGT_FLAG_EWIF);
```



## wwdgt\_flag\_clear

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

**Table 3-1044. Function wwdgt\_flag\_clear**

<b>Function name</b>	wwdgt_flag_clear
<b>Function prototype</b>	void wwdgt_flag_clear(uint32_t flag);
<b>Function descriptions</b>	clear flag state of WWDGT
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	flag state of WWDGT
WWDGT_FLAG_EWIF	early wakeup interrupt flag
WWDGT_FLAG_REFE F	refresh error flag
WWDGT_FLAG_UNDF F	underflow error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear early wakeup interrupt state of WWDGT */
```

```
wwdgt_flag_clear(WWDGT_FLAG_EWIF);
```

## 4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial version	Feb.28, 2026

## Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company according to the laws of the People's Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.