

**GigaDevice Semiconductor Inc.**

**GD32H75E**

**Arm<sup>®</sup> Cortex<sup>®</sup>-M7 32-bit MCU**

**Firmware Library  
User Guide**

Revision 1.2

(Aug. 2025)

# Table of Contents

<b>Table of Contents .....</b>	<b>1</b>
<b>List of Figures .....</b>	<b>5</b>
<b>List of Tables .....</b>	<b>6</b>
<b>1. Introduction .....</b>	<b>40</b>
<b>1.1. Rules of User Manual and Firmware Library .....</b>	<b>40</b>
1.1.1. Peripherals.....	40
1.1.2. Naming rules.....	41
<b>2. Firmware Library Overview.....</b>	<b>43</b>
<b>2.1. File Structure of Firmware Library .....</b>	<b>43</b>
2.1.1. Examples Folder .....	45
2.1.2. Firmware Folder.....	45
2.1.3. Template Folder .....	45
2.1.4. Utilities Folder .....	47
<b>2.2. File descriptions of Firmware Library .....</b>	<b>47</b>
<b>3. Firmware Library of Standard Peripherals .....</b>	<b>49</b>
<b>3.1. Overview of Firmware Library of Standard Peripherals.....</b>	<b>49</b>
<b>3.2. ADC .....</b>	<b>49</b>
3.2.1. Descriptions of Peripheral registers.....	49
3.2.2. Descriptions of Peripheral functions.....	50
<b>3.3. CAN .....</b>	<b>89</b>
3.3.1. Descriptions of Peripheral registers.....	89
3.3.2. Descriptions of Peripheral functions .....	90
<b>3.4. CMP .....</b>	<b>133</b>
3.4.1. Descriptions of Peripheral registers.....	134
3.4.2. Descriptions of Peripheral functions .....	134
<b>3.5. CPDM.....</b>	<b>148</b>
3.5.1. Descriptions of Peripheral registers.....	148
3.5.2. Descriptions of Peripheral functions .....	148
<b>3.6. CRC .....</b>	<b>154</b>
3.6.1. Descriptions of Peripheral registers.....	154
3.6.2. Descriptions of Peripheral functions .....	154
<b>3.7. CTC.....</b>	<b>162</b>
3.7.1. Descriptions of Peripheral registers.....	162
3.7.2. Descriptions of Peripheral functions .....	162

<b>3.8. DAC</b>	<b>175</b>
3.8.1. Peripheral register description	175
3.8.2. Descriptions of Peripheral functions	176
<b>3.9. DBG</b>	<b>196</b>
3.9.1. Descriptions of Peripheral registers	196
3.9.2. Descriptions of Peripheral functions	196
<b>3.10. DMA / DMAMUX</b>	<b>202</b>
3.10.1. Descriptions of Peripheral registers	202
3.10.2. Descriptions of Peripheral functions	203
<b>3.11. EDOUT</b>	<b>258</b>
3.11.1. Descriptions of Peripheral registers	258
3.11.2. Descriptions of Peripheral functions	259
<b>3.12. EFUSE</b>	<b>265</b>
3.12.1. Descriptions of Peripheral registers	265
3.12.2. Descriptions of Peripheral functions	265
<b>3.13. EXMC</b>	<b>278</b>
3.13.1. Descriptions of Peripheral registers	278
3.13.2. Descriptions of Peripheral functions	278
<b>3.14. EXTI</b>	<b>304</b>
3.14.1. Descriptions of Peripheral registers	304
3.14.2. Descriptions of Peripheral functions	305
<b>3.15. FAC</b>	<b>312</b>
3.15.1. Descriptions of Peripheral registers	312
3.15.2. Descriptions of Peripheral functions	313
<b>3.16. FMC</b>	<b>331</b>
3.16.1. Descriptions of Peripheral registers	331
3.16.2. Descriptions of Peripheral functions	332
<b>3.17. FWDGT</b>	<b>367</b>
3.17.1. Descriptions of Peripheral registers	368
3.17.2. Descriptions of Peripheral functions	368
<b>3.18. GPIO</b>	<b>373</b>
3.18.1. Descriptions of Peripheral registers	373
3.18.2. Descriptions of Peripheral functions	374
<b>3.19. HPDF</b>	<b>385</b>
3.19.1. Descriptions of Peripheral registers	385
3.19.2. Descriptions of Peripheral functions	386
<b>3.20. I2C</b>	<b>443</b>
3.20.1. Descriptions of Peripheral registers	443
3.20.2. Descriptions of Peripheral functions	444

<b>3.21.</b>	<b>LPDTS</b> .....	<b>482</b>
3.21.1.	Descriptions of Peripheral registers.....	482
3.21.2.	Descriptions of Peripheral functions .....	482
<b>3.22.</b>	<b>MDMA</b> .....	<b>492</b>
3.22.1.	Descriptions of Peripheral registers.....	492
3.22.2.	Descriptions of Peripheral functions .....	493
<b>3.23.</b>	<b>OSPI</b> .....	<b>522</b>
3.23.1.	Descriptions of Peripheral registers.....	522
3.23.2.	Descriptions of Peripheral functions .....	523
<b>3.24.</b>	<b>OSPIM</b> .....	<b>562</b>
3.24.1.	Descriptions of Peripheral registers.....	562
3.24.2.	Descriptions of Peripheral functions .....	562
<b>3.25.</b>	<b>MISC</b> .....	<b>567</b>
3.25.1.	Descriptions of Peripheral registers.....	567
3.25.2.	Descriptions of Peripheral functions .....	568
<b>3.26.</b>	<b>PMU</b> .....	<b>579</b>
3.26.1.	Descriptions of Peripheral registers.....	579
3.26.2.	Descriptions of Peripheral functions .....	579
<b>3.27.</b>	<b>RAMECCMU</b> .....	<b>597</b>
3.27.1.	Descriptions of Peripheral registers.....	598
3.27.2.	Descriptions of Peripheral functions .....	598
<b>3.28.</b>	<b>RCU</b> .....	<b>612</b>
3.28.1.	Descriptions of Peripheral registers.....	612
3.28.2.	Descriptions of Peripheral functions .....	613
<b>3.29.</b>	<b>RTC</b> .....	<b>667</b>
3.29.1.	Descriptions of Peripheral registers.....	667
3.29.2.	Descriptions of Peripheral functions .....	669
<b>3.30.</b>	<b>SPI</b> .....	<b>694</b>
3.30.1.	Descriptions of Peripheral registers.....	694
3.30.2.	Descriptions of Peripheral functions .....	695
<b>3.31.</b>	<b>SYSCFG</b> .....	<b>738</b>
3.31.1.	Descriptions of Peripheral registers.....	738
3.31.2.	Descriptions of Peripheral functions .....	739
<b>3.32.</b>	<b>TIMER</b> .....	<b>760</b>
3.32.1.	Descriptions of Peripheral registers.....	761
3.32.2.	Descriptions of Peripheral functions .....	762
<b>3.33.</b>	<b>TMU</b> .....	<b>849</b>
3.33.1.	Descriptions of Peripheral registers.....	849
3.33.2.	Descriptions of Peripheral functions .....	850



<b>3.34. TRIGSEL .....</b>	<b>859</b>
3.34.1. Descriptions of Peripheral registers .....	859
3.34.2. Descriptions of Peripheral functions .....	860
<b>3.35. TRNG.....</b>	<b>868</b>
3.35.1. Descriptions of Peripheral registers .....	868
3.35.2. Descriptions of Peripheral functions .....	868
<b>3.36. USART.....</b>	<b>889</b>
3.36.1. Descriptions of Peripheral registers .....	889
3.36.2. Descriptions of Peripheral functions .....	889
<b>3.37. VREF .....</b>	<b>941</b>
3.37.1. Descriptions of Peripheral registers .....	941
3.37.2. Descriptions of Peripheral functions .....	941
<b>3.38. WWDGT.....</b>	<b>946</b>
3.38.1. Descriptions of Peripheral registers .....	946
3.38.2. Descriptions of Peripheral functions .....	946
<b>3.39. ESC_INTC .....</b>	<b>951</b>
3.39.1. Descriptions of Peripheral registers .....	951
3.39.2. Descriptions of Peripheral functions .....	951
<b>3.40. ESC_OSPI.....</b>	<b>958</b>
3.40.1. Descriptions of Peripheral registers .....	958
3.40.2. Descriptions of Peripheral functions .....	958
<b>3.41. ESC_PHY .....</b>	<b>968</b>
3.41.1. Descriptions of Peripheral registers .....	968
3.41.2. Descriptions of Peripheral functions .....	968
<b>3.42. ESC_PMU.....</b>	<b>971</b>
3.42.1. Descriptions of Peripheral registers .....	971
3.42.2. Descriptions of Peripheral functions .....	971
<b>3.43. ESC_SYSCFG .....</b>	<b>978</b>
3.43.1. Descriptions of Peripheral registers .....	978
3.43.2. Descriptions of Peripheral functions .....	979
<b>3.44. ESC_TIMER .....</b>	<b>981</b>
3.44.1. Descriptions of Peripheral registers .....	981
3.44.2. Descriptions of Peripheral functions .....	981
<b>4. Revision history .....</b>	<b>985</b>

## List of Figures

Figure 2-1. File structure of firmware library of GD32H75E.....	44
Figure 2-2. Select peripheral example files .....	46
Figure 2-3. Copy the peripheral example files .....	46
Figure 2-4. Open the project file .....	46
Figure 2-5. Configure project files .....	47
Figure 2-6. Compile-debug-download .....	47

# List of Tables

Table 1-1. Peripherals .....	40
Table 2-1. Function descriptions of Firmware Library .....	48
Table 3-1. Peripheral function format of Firmware Library .....	49
Table 3-2. ADC Registers .....	49
Table 3-3. ADC firmware function.....	50
Table 3-4. Function adc_deinit.....	52
Table 3-5. Function adc_clock_config.....	52
Table 3-6. Function adc_special_function_config.....	54
Table 3-7. Function adc_data_alignment_config.....	54
Table 3-8. Function adc_enable .....	55
Table 3-9. Function adc_disable .....	56
Table 3-10. Function adc_calibration_mode_config .....	56
Table 3-11. Function adc_calibration_number .....	57
Table 3-12. Function adc_calibration_enable.....	58
Table 3-13. Function adc_resolution_config .....	58
Table 3-14. Function adc_internal_channel_config .....	59
Table 3-15. Function adc_dma_mode_enable.....	60
Table 3-16. Function adc_dma_mode_disable.....	60
Table 3-17. Function adc_dma_request_after_last_enable .....	61
Table 3-18. Function adc_dma_request_after_last_disable .....	61
Table 3-19. Function adc_hpdf_mode_enable .....	62
Table 3-20. Function adc_hpdf_mode_disable .....	62
Table 3-21. Function adc_discontinuous_mode_config .....	63
Table 3-22. Function adc_channel_length_config.....	64
Table 3-23. Function adc_routine_channel_config .....	64
Table 3-24. Function adc_inserted_channel_config .....	65
Table 3-25. Function adc_inserted_channel_offset_config.....	66
Table 3-26. Function adc_channel_differential_mode_config .....	66
Table 3-27. Function adc_external_trigger_config.....	67
Table 3-28. Function adc_software_trigger_enable .....	68
Table 3-29. Function adc_end_of_conversion_config.....	69
Table 3-30. Function adc_routine_data_read .....	69
Table 3-31. Function adc_inserted_data_read .....	70
Table 3-32. Function adc_watchdog0_single_channel_enable .....	71
Table 3-33. Function adc_watchdog0_group_channel_enable.....	71
Table 3-34. Function adc_watchdog0_disable .....	72
Table 3-35. Function adc_watchdog1_channel_config .....	72
Table 3-36. Function adc_watchdog2_channel_config .....	73
Table 3-37. Function adc_watchdog1_disable .....	74
Table 3-38. Function adc_watchdog2_disable .....	74
Table 3-39. Function adc_watchdog0_threshold_config .....	75

Table 3-40. Function adc_watchdog1_threshold_config .....	76
Table 3-41. Function adc_watchdog2_threshold_config .....	76
Table 3-42. Function adc_oversample_mode_config .....	77
Table 3-43. Function adc_oversample_mode_enable .....	78
Table 3-44. Function adc_oversample_mode_disable .....	79
Table 3-45. Function adc_flag_get .....	79
Table 3-46. Function adc_flag_clear .....	80
Table 3-47. Function adc_interrupt_enable .....	81
Table 3-48. Function adc_interrupt_disable .....	81
Table 3-49. Function adc_interrupt_flag_get.....	82
Table 3-50. Function adc_interrupt_flag_clear .....	83
Table 3-51. Function adc_sync_mode_config.....	84
Table 3-52. Function adc_interrupt_disable .....	85
Table 3-53. Function adc_sync_dma_config .....	85
Table 3-54. Function adc_sync_dma_request_after_last_enable .....	86
Table 3-55. Function adc_sync_dma_request_after_last_disable .....	87
Table 3-56. Function adc_sync_master_adc_routine_data0_read .....	87
Table 3-57. Function adc_sync_slave_adc_routine_data0_read .....	88
Table 3-58. Function adc_sync_routine_data1_read .....	88
Table 3-59. CAN Registers .....	89
Table 3-60. CAN firmware function .....	90
Table 3-61. Structure can_error_counter_struct.....	91
Table 3-62. Structure can_parameter_struct .....	92
Table 3-63. Structure can_mailbox_descriptor_struct .....	92
Table 3-64. Structure can_rx_fifo_struct.....	93
Table 3-65. Structure can_fd_parameter_struct .....	93
Table 3-66. Structure can_rx_fifo_id_filter_struct .....	93
Table 3-67. Structure can_fifo_parameter_struct .....	93
Table 3-68. Structure can_pn_mode_filter_struct .....	94
Table 3-69. Structure can_pn_mode_config_struct .....	94
Table 3-70. Structure can_crc_struct.....	94
Table 3-71. Enum can_interrupt_enum.....	95
Table 3-72. Enum can_flag_enum .....	96
Table 3-73. Enum can_interrupt_flag_enum.....	98
Table 3-74. Enum can_operation_modes_enum.....	100
Table 3-75. Enum can_struct_type_enum.....	101
Table 3-76. Enum can_error_state_enum .....	101
Table 3-77. Function can_deinit.....	101
Table 3-78. Function can_software_reset .....	102
Table 3-79. Function can_init .....	102
Table 3-80. Function can_struct_para_init .....	103
Table 3-81. Function can_private_filter_config.....	104
Table 3-82. Function can_operation_mode_enter .....	104
Table 3-83. Function can_operation_mode_get.....	105

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

Table 3-128. Function can_flag_get .....	130
Table 3-129. Function can_flag_clear .....	131
Table 3-130. Function can_interrupt_enable .....	131
Table 3-131. Function can_interrupt_disable .....	132
Table 3-132. Function can_interrupt_flag_get.....	132
Table 3-133. Function can_interrupt_flag_clear .....	133
Table 3-134. CMP registers .....	134
Table 3-135. CMP firmware function .....	134
Table 3-136. Enum cmp_enum .....	135
Table 3-137. Function cmp_deinit .....	135
Table 3-138. Function cmp_mode_init.....	135
Table 3-139. Function cmp_noninverting_input_select .....	137
Table 3-140. Function cmp_output_init.....	137
Table 3-141. Function cmp_output_mux_config .....	138
Table 3-142. Function cmp_outputblank_init.....	139
Table 3-143. Function cmp_enable.....	140
Table 3-144. Function cmp_disable .....	140
Table 3-145. Function cmp_window_enable.....	141
Table 3-146. Function cmp_window_disable .....	141
Table 3-147. Function cmp_lock_enable.....	142
Table 3-148. Function cmp_voltage_scaler_enable .....	142
Table 3-149. Function cmp_voltage_scaler_disable .....	143
Table 3-150. Function cmp_scaler_bridge_enable.....	143
Table 3-151. Function cmp_scaler_bridge_disable.....	144
Table 3-152. Function cmp_output_level_get .....	144
Table 3-153. Function cmp_flag_get.....	145
Table 3-154. Function cmp_flag_clear.....	145
Table 3-155. Function cmp_interrupt_enable.....	146
Table 3-156. Function cmp_interrupt_disable.....	146
Table 3-157. Function cmp_interrupt_flag_get .....	147
Table 3-158. Function cmp_interrupt_flag_clear .....	147
Table 3-159. CPDM Registers .....	148
Table 3-160. CPDM firmware function .....	148
Table 3-161. Enum cpdm_output_phase_enum.....	149
Table 3-162. Function cpdm_enable .....	150
Table 3-163. Function cpdm_disable .....	150
Table 3-164. Function cpdm_delayline_sample_enable.....	151
Table 3-165. Function cpdm_delayline_sample_disable.....	151
Table 3-166. Function cpdm_output_clock_phase_select.....	152
Table 3-167. Function cpdm_delayline_length_valid_flag_get .....	152
Table 3-168. Function cpdm_delayline_length_get.....	153
Table 3-169. Function cpdm_clock_output.....	153
Table 3-170. CRC Registers .....	154
Table 3-171. CRC firmware function .....	154

Table 3-172. Function <code>crc_deinit</code> .....	155
Table 3-173. Function <code>crc_init_data_register_write</code> .....	155
Table 3-174. Function <code>crc_data_register_read</code> .....	156
Table 3-175. Function <code>crc_free_data_register_read</code> .....	156
Table 3-176. Function <code>crc_free_data_register_write</code> .....	157
Table 3-177. Function <code>crc_reverse_output_data_disable</code> .....	157
Table 3-178. Function <code>crc_reverse_output_data_enable</code> .....	158
Table 3-179. Function <code>crc_input_data_reverse_config</code> .....	158
Table 3-180. Function <code>crc_data_register_reset</code> .....	159
Table 3-181. Function <code>crc_polynomial_size_set</code> .....	159
Table 3-182. Function <code>crc_polynomial_set</code> .....	160
Table 3-183. Function <code>crc_single_data_calculate</code> .....	160
Table 3-184. Function <code>crc_block_data_calculate</code> .....	161
Table 3-185. CTC Registers .....	162
Table 3-186. CTC firmware function.....	163
Table 3-187. Function <code>ctc_deinit</code> .....	163
Table 3-188. Function <code>ctc_counter_enable</code> .....	164
Table 3-189. Function <code>ctc_counter_disable</code> .....	164
Table 3-190. Function <code>ctc_irc48m_trim_value_config</code> .....	165
Table 3-191. Function <code>ctc_software_refsource_pulse_generate</code> .....	165
Table 3-192. Function <code>ctc_hardware_trim_mode_config</code> .....	166
Table 3-193. Function <code>ctc_refsource_polarity_config</code> .....	166
Table 3-194. Function <code>ctc_refsource_signal_select</code> .....	167
Table 3-195. Function <code>ctc_refsource_prescaler_config</code> .....	167
Table 3-196. Function <code>ctc_clock_limit_value_config</code> .....	168
Table 3-197. Function <code>ctc_counter_reload_value_config</code> .....	169
Table 3-198. Function <code>ctc_counter_capture_value_read</code> .....	169
Table 3-199. Function <code>ctc_counter_direction_read</code> .....	170
Table 3-200. Function <code>ctc_counter_reload_value_read</code> .....	170
Table 3-201. Function <code>ctc_irc48m_trim_value_read</code> .....	171
Table 3-202. Function <code>ctc_flag_get</code> .....	171
Table 3-203. Function <code>ctc_flag_clear</code> .....	172
Table 3-204. Function <code>ctc_interrupt_enable</code> .....	172
Table 3-205. Function <code>ctc_interrupt_disable</code> .....	173
Table 3-206. Function <code>ctc_interrupt_flag_get</code> .....	173
Table 3-207. Function <code>ctc_interrupt_flag_clear</code> .....	174
Table 3-208. DAC Registers .....	175
Table 3-209. DAC firmware functions .....	176
Table 3-210. Function <code>dac_deinit</code> .....	177
Table 3-211. Function <code>dac_enable</code> .....	177
Table 3-212. Function <code>dac_disable</code> .....	178
Table 3-213. Function <code>dac_dma_enable</code> .....	178
Table 3-214. Function <code>dac_dma_disable</code> .....	179
Table 3-215. Function <code>dac_mode_config</code> .....	179

Table 3-216. Function <code>dac_trimming_value_get</code> .....	180
Table 3-217. Function <code>dac_trimming_value_set</code> .....	181
Table 3-218. Function <code>dac_trimming_enable</code> .....	182
Table 3-219. Function <code>dac_output_value_get</code> .....	182
Table 3-220. Function <code>dac_data_set</code> .....	183
Table 3-221. Function <code>dac_trigger_enable</code> .....	183
Table 3-222. Function <code>dac_trigger_disable</code> .....	184
Table 3-223. Function <code>dac_trigger_source_config</code> .....	185
Table 3-224. Function <code>dac_software_trigger_enable</code> .....	185
Table 3-225. Function <code>dac_wave_mode_config</code> .....	186
Table 3-226. Function <code>dac_lfsr_noise_config</code> .....	187
Table 3-227. Function <code>dac_triangle_noise_config</code> .....	187
Table 3-228. Function <code>dac_concurrent_enable</code> .....	188
Table 3-229. Function <code>dac_concurrent_disable</code> .....	189
Table 3-230. Function <code>dac_concurrent_software_trigger_enable</code> .....	189
Table 3-231. Function <code>dac_concurrent_data_set</code> .....	190
Table 3-232. Function <code>dac_sample_keep_mode_config</code> .....	190
Table 3-233. Function <code>dac_flag_get</code> .....	191
Table 3-234. Function <code>dac_flag_clear</code> .....	192
Table 3-235. Function <code>dac_interrupt_enable</code> .....	192
Table 3-236. Function <code>dac_interrupt_disable</code> .....	193
Table 3-237. Function <code>dac_interrupt_flag_get</code> .....	193
Table 3-238. Function <code>dac_interrupt_flag_clear</code> .....	194
Table 3-239. DBG Registers .....	196
Table 3-240. DBG firmware function .....	196
Table 3-241. Enum <code>dbg_periph_enum</code> .....	196
Table 3-242. Function <code>dbg_deinit</code> .....	197
Table 3-243. Function <code>dbg_id_get</code> .....	197
Table 3-244. Function <code>dbg_low_power_enable</code> .....	198
Table 3-245. Function <code>dbg_low_power_disable</code> .....	198
Table 3-246. Function <code>dbg_periph_enable</code> .....	199
Table 3-247. Function <code>dbg_periph_disable</code> .....	200
Table 3-248. Function <code>dbg_trace_pin_enable</code> .....	200
Table 3-249. Function <code>dbg_trace_pin_disable</code> .....	201
Table 3-250. Function <code>dbg_trace_pin_mode_set</code> .....	201
Table 3-251. DMA Registers .....	202
Table 3-252. DMAMUX Registers .....	203
Table 3-253. DMA firmware function .....	203
Table 3-254. DMAMUX firmware function .....	204
Table 3-255. Structure <code>dma_multi_data_parameter_struct</code> .....	205
Table 3-256. Structure <code>dma_single_data_parameter_struct</code> .....	205
Table 3-257. Structure <code>dmamux_sync_parameter_struct</code> .....	206
Table 3-258. Structure <code>dmamux_gen_parameter_struct</code> .....	206
Table 3-259. Enum <code>dma_channel_enum</code> .....	206



Table 3-260. Enum dmamux_muxlexer_channel_enum .....	207
Table 3-261. Enum dmamux_generator_channel_enum .....	207
Table 3-262. Enum dmamux_interrupt_enum .....	208
Table 3-263. Enum dmamux_flag_enum .....	209
Table 3-264. Enum dmamux_interrupt_flag_enum.....	210
Table 3-265. Function dma_deinit .....	211
Table 3-266. Function dma_single_data_para_struct_init .....	212
Table 3-267. Function dma_multi_data_para_struct_init .....	212
Table 3-268. Function dma_single_data_mode_init.....	213
Table 3-269. Function dma_multi_data_mode_init.....	214
Table 3-270. Function dma_periph_address_config .....	215
Table 3-271. Function dma_memory_address_config.....	216
Table 3-272. Function dma_transfer_number_config .....	217
Table 3-273. Function dma_transfer_number_get.....	217
Table 3-274. Function dma_priority_config .....	218
Table 3-275. Function dma_memory_burst_beats_config.....	219
Table 3-276. Function dma_periph_burst_beats_config.....	219
Table 3-277. Function dma_memory_width_config .....	220
Table 3-278. Function dma_periph_width_config .....	221
Table 3-279. Function dma_memory_address_generation_config .....	222
Table 3-280. Function dma_peripheral_address_generation_config.....	222
Table 3-281. Function dma_circulation_enable .....	223
Table 3-282. Function dma_circulation_disable .....	224
Table 3-283. Function dma_channel_enable .....	224
Table 3-284. Function dma_channel_disable .....	225
Table 3-285. Function dma_transfer_direction_config.....	225
Table 3-286. Function dma_switch_buffer_mode_config .....	226
Table 3-287. Function dma_using_memory_get.....	227
Table 3-288. Function dma_switch_buffer_mode_enable.....	228
Table 3-289. Function dma_switch_buffer_mode_disable.....	228
Table 3-290. Function dma_fifo_status_get.....	229
Table 3-291. Function dma_flag_get .....	230
Table 3-292. Function dma_flag_clear .....	230
Table 3-293. Function dma_interrupt_enable.....	231
Table 3-294. Function dma_interrupt_disable.....	232
Table 3-295. Function dma_interrupt_flag_get .....	233
Table 3-296. Function dma_interrupt_flag_clear .....	233
Table 3-297. Function dmamux_sync_struct_para_init.....	234
Table 3-298. Function dmamux_synchronization_init.....	235
Table 3-299. Function dmamux_synchronization_enable.....	235
Table 3-300. Function dmamux_synchronization_disable.....	236
Table 3-301. Function dmamux_event_generation_enable .....	236
Table 3-302. Function dmamux_event_generation_disable .....	237
Table 3-303. Function dmamux_gen_struct_para_init .....	238

Table 3-304. Function <code>dmamux_request_generator_init</code> .....	238
Table 3-305. Function <code>dmamux_request_generator_channel_enable</code> .....	239
Table 3-306. Function <code>dmamux_request_generator_channel_disable</code> .....	239
Table 3-307. Function <code>dmamux_synchronization_polarity_config</code> .....	240
Table 3-308. Function <code>dmamux_request_forward_number_config</code> .....	241
Table 3-309. Function <code>dmamux_sync_id_config</code> .....	241
Table 3-310. Function <code>dmamux_request_id_config</code> .....	243
Table 3-311. Function <code>dmamux_trigger_polarity_config</code> .....	252
Table 3-312. Function <code>dmamux_request_generate_number_config</code> .....	252
Table 3-313. Function <code>dmamux_trigger_id_config</code> .....	253
Table 3-314. Function <code>dmamux_flag_get</code> .....	255
Table 3-315. Function <code>dmamux_flag_clear</code> .....	256
Table 3-316. Function <code>dmamux_interrupt_enable</code> .....	256
Table 3-317. Function <code>dmamux_interrupt_disable</code> .....	257
Table 3-318. Function <code>dmamux_interrupt_flag_get</code> .....	257
Table 3-319. Function <code>dmamux_interrupt_flag_clear</code> .....	258
Table 3-320. EDOUT Registers .....	258
Table 3-321. EDOUT firmware function .....	259
Table 3-322. Function <code>edout_deinit</code> .....	259
Table 3-323. Function <code>edout_init</code> .....	260
Table 3-324. Function <code>edout_enable</code> .....	260
Table 3-325. Function <code>edout_disable</code> .....	261
Table 3-326. Function <code>edout_polarity_config</code> .....	261
Table 3-327. Function <code>edout_max_location_value_config</code> .....	262
Table 3-328. Function <code>edout_output_counter_update</code> .....	262
Table 3-329. Function <code>edout_current_location_config</code> .....	263
Table 3-330. Function <code>edout_current_location_get</code> .....	263
Table 3-331. Function <code>edout_z_output_mode_config</code> .....	264
Table 3-332. Function <code>edout_z_output_start_loc_and_width_config</code> .....	264
Table 3-333. EFUSE Registers .....	265
Table 3-334. EFUSE firmware function .....	265
Table 3-335. Enum <code>efuse_system_para_size_enum</code> .....	266
Table 3-336. Enum <code>efuse_system_para_index_enum</code> .....	266
Table 3-337. Enum <code>efuse_state_enum</code> .....	267
Table 3-338. Enum <code>efuse_interrupt_flag_enum</code> .....	267
Table 3-339. Function <code>efuse_read</code> .....	267
Table 3-340. Function <code>efuse_program</code> .....	268
Table 3-341. Function <code>efuse_user_control_write</code> .....	269
Table 3-342. Function <code>efuse_mcu_reserved_write</code> .....	269
Table 3-343. Function <code>efuse_dp_write</code> .....	270
Table 3-344. Function <code>efuse_aes_key_write</code> .....	270
Table 3-345. Function <code>efuse_user_data_write</code> .....	271
Table 3-346. Function <code>efuse_aes_key_crc_get</code> .....	271
Table 3-347. Function <code>efuse_monitor_program_voltage_enable</code> .....	272

Table 3-348. Function <code>efuse_monitor_program_voltage_disable</code> .....	272
Table 3-349. Function <code>efuse_monitor_program_voltage_get</code> .....	273
Table 3-350. Function <code>efuse_ldo_ready_get</code> .....	273
Table 3-351. Function <code>efuse_flag_get</code> .....	274
Table 3-352. Function <code>efuse_flag_clear</code> .....	274
Table 3-353. Function <code>efuse_interrupt_enable</code> .....	275
Table 3-354. Function <code>efuse_interrupt_disable</code> .....	276
Table 3-355. Function <code>efuse_interrupt_flag_get</code> .....	276
Table 3-356. Function <code>efuse_interrupt_flag_clear</code> .....	277
Table 3-357. EXMC Registers .....	278
Table 3-358. EXMC firmware function.....	279
Table 3-359. Structure <code>exmc_norsram_timing_parameter_struct</code> .....	280
Table 3-360. Structure <code>exmc_norsram_parameter_struct</code> .....	280
Table 3-361. Structure <code>exmc_nand_timing_parameter_struct</code> .....	280
Table 3-362. Structure <code>exmc_nand_parameter_struct</code> .....	281
Table 3-363. Structure <code>exmc_sdram_timing_parameter_struct</code> .....	281
Table 3-364. Structure <code>exmc_sdram_parameter_struct</code> .....	281
Table 3-365. Structure <code>exmc_sdram_command_parameter_struct</code> .....	282
Table 3-366. Function <code>exmc_norsram_deinit</code> .....	282
Table 3-367. Function <code>exmc_norsram_struct_para_init</code> .....	283
Table 3-368. Function <code>exmc_norsram_init</code> .....	283
Table 3-369. Function <code>exmc_norsram_enable</code> .....	285
Table 3-370. Function <code>exmc_norsram_disable</code> .....	285
Table 3-371. Function <code>exmc_nand_deinit</code> .....	286
Table 3-372. Function <code>exmc_nand_struct_para_init</code> .....	286
Table 3-373. Function <code>exmc_nand_init</code> .....	287
Table 3-374. Function <code>exmc_nand_enable</code> .....	288
Table 3-375. Function <code>exmc_nand_disable</code> .....	288
Table 3-376. Function <code>exmc_sdram_deinit</code> .....	289
Table 3-377. Function <code>exmc_sdram_struct_para_init</code> .....	289
Table 3-378. Function <code>exmc_sdram_init</code> .....	290
Table 3-379. Function <code>exmc_norsram_sdram_remap_config</code> .....	291
Table 3-380. Function <code>exmc_norsram_sdram_remap_get</code> .....	292
Table 3-381. Function <code>exmc_norsram_consecutive_clock_config</code> .....	292
Table 3-382. Function <code>exmc_norsram_page_size_config</code> .....	293
Table 3-383. Function <code>exmc_nand_ecc_config</code> .....	294
Table 3-384. Function <code>exmc_ecc_get</code> .....	294
Table 3-385. Function <code>exmc_sdram_readsample_enable</code> .....	295
Table 3-386. Function <code>exmc_sdram_readsample_disable</code> .....	295
Table 3-387. Function <code>exmc_sdram_readsample_config</code> .....	296
Table 3-388. Function <code>exmc_sdram_command_config</code> .....	296
Table 3-389. Function <code>exmc_sdram_refresh_count_set</code> .....	297
Table 3-390. Function <code>exmc_sdram_autorefresh_number_set</code> .....	297
Table 3-391. Function <code>exmc_sdram_write_protection_config</code> .....	298

Table 3-392. Function <code>exmc_sdram_bankstatus_get</code> .....	298
Table 3-393. Function <code>exmc_flag_get</code> .....	299
Table 3-394. Function <code>exmc_flag_clear</code> .....	300
Table 3-395. Function <code>exmc_interrupt_enable</code> .....	301
Table 3-396. Function <code>exmc_interrupt_disable</code> .....	302
Table 3-397. Function <code>exmc_interrupt_flag_get</code> .....	302
Table 3-398. Function <code>exmc_interrupt_flag_clear</code> .....	303
Table 3-399. EXTI Registers.....	304
Table 3-400. EXTI firmware function .....	305
Table 3-401. Enum <code>exti_line_enum</code> .....	305
Table 3-402. Enum <code>exti_mode_enum</code> .....	306
Table 3-403. Enum <code>exti_trig_type_enum</code> .....	306
Table 3-404. Function <code>exti_deinit</code> .....	306
Table 3-405. Function <code>exti_init</code> .....	307
Table 3-406. Function <code>exti_interrupt_enable</code> .....	308
Table 3-407. Function <code>exti_interrupt_disable</code> .....	308
Table 3-408. Function <code>exti_event_enable</code> .....	308
Table 3-409. Function <code>exti_event_disable</code> .....	309
Table 3-410. Function <code>exti_software_interrupt_enable</code> .....	309
Table 3-411. Function <code>exti_software_interrupt_disable</code> .....	310
Table 3-412. Function <code>exti_flag_get</code> .....	310
Table 3-413. Function <code>exti_flag_clear</code> .....	311
Table 3-414. Function <code>exti_interrupt_flag_get</code> .....	311
Table 3-415. Function <code>exti_interrupt_flag_clear</code> .....	312
Table 3-416. FAC Registers.....	312
Table 3-417. FAC firmware function .....	313
Table 3-418. Structure <code>fac_parameter_struct</code> .....	314
Table 3-419. Structure <code>fac_fixed_data_preload_struct</code> .....	314
Table 3-420. Structure <code>fac_float_data_preload_struct</code> .....	315
Table 3-421. Function <code>fac_deinit</code> .....	315
Table 3-422. Function <code>fac_struct_para_init</code> .....	315
Table 3-423. Function <code>fac_fixed_data_preload_init</code> .....	316
Table 3-424. Function <code>fac_float_data_preload_init</code> .....	316
Table 3-425. Function <code>fac_init</code> .....	317
Table 3-426. Function <code>fac_preload</code> .....	318
Table 3-427. Function <code>fac_float_buffer_preload</code> .....	318
Table 3-428. Function <code>fac_float_preload</code> .....	319
Table 3-429. Function <code>fac_float_preload</code> .....	319
Table 3-430. Function <code>fac_init</code> .....	320
Table 3-431. Function <code>fac_clip_config</code> .....	320
Table 3-432. Function <code>fac_init</code> .....	321
Table 3-433. Function <code>fac_float_disable</code> .....	321
Table 3-434. Function <code>fac_dma_enable</code> .....	322
Table 3-435. Function <code>fac_dma_enable</code> .....	322

Table 3-436. Function fac_dma_enable .....	323
Table 3-437. Function fac_dma_enable .....	323
Table 3-438. Function fac_dma_enable .....	324
Table 3-439. Function fac_dma_enable .....	324
Table 3-440. Function fac_start.....	325
Table 3-441. Function fac_start.....	326
Table 3-442. Function fac_start.....	326
Table 3-443. Function fac_fixed_data_write .....	326
Table 3-444. Function fac_fixed_data_write .....	327
Table 3-445. Function fac_fixed_data_write .....	327
Table 3-446. Function fac_fixed_data_write .....	328
Table 3-447. Function fac_interrupt_enable .....	328
Table 3-448. Function fac_interrupt_disable .....	329
Table 3-449. Function fac_interrupt_flag_get.....	330
Table 3-450. Function fac_flag_get .....	330
Table 3-451. FMC Registers .....	331
Table 3-452. FMC firmware function .....	332
Table 3-453. Enum fmc_state_enum .....	333
Table 3-454. Enum fmc_flag_enum .....	334
Table 3-455. Enum fmc_interrupt_flag_enum .....	334
Table 3-456. Enum fmc_interrupt_enum .....	335
Table 3-457. Function fmc_unlock .....	335
Table 3-458. Function fmc_lock .....	336
Table 3-459. Function fmc_page_erase.....	336
Table 3-460. Function fmc_mass_erase.....	337
Table 3-461. Function fmc_protection_removed_mass_erase .....	337
Table 3-462. Function fmc_word_program .....	338
Table 3-463. Function fmc_doubleword_program.....	338
Table 3-464. Function fmc_check_programming_area_enable .....	339
Table 3-465. Function fmc_check_programming_area_disable .....	339
Table 3-466. Function ob_unlock .....	340
Table 3-467. Function ob_lock .....	340
Table 3-468. Function ob_start .....	341
Table 3-469. Function ob_factory_value_config.....	341
Table 3-470. Function ob_secure_access_mode_enable .....	342
Table 3-471. Function ob_secure_access_mode_disable .....	343
Table 3-472. Function ob_security_protection_config .....	343
Table 3-473. Function ob_bor_threshold_config.....	344
Table 3-474. Function ob_low_power_config.....	344
Table 3-475. Function ob_tcm_ecc_config .....	346
Table 3-476. Function ob_iospeed_optimize_config .....	347
Table 3-477. Function ob_tcm_shared_ram_config.....	347
Table 3-478. Function ob_data_program .....	349
Table 3-479. Function ob_boot_address_config .....	349

Table 3-480. Function ob_dcrp_area_config .....	350
Table 3-481. Function ob_secure_area_config .....	351
Table 3-482. Function ob_write_protection_enable .....	351
Table 3-483. Function ob_write_protection_disable .....	352
Table 3-484. Function ob_secure_mode_get .....	354
Table 3-485. Function ob_secure_mode_get .....	354
Table 3-486. Function ob_bor_threshold_get .....	355
Table 3-487. Function ob_low_power_get .....	355
Table 3-488. Function ob_tcm_ecc_get.....	356
Table 3-489. Function ob_secure_mode_get .....	357
Table 3-490. Function ob_itcm_shared_ram_size_get .....	358
Table 3-491. Function ob_dtcn_shared_ram_size_get .....	358
Table 3-492. Function ob_data_get .....	359
Table 3-493. Function ob_boot_address_get .....	359
Table 3-494. Function ob_dcrp_area_get.....	360
Table 3-495. Function ob_secure_area_get.....	360
Table 3-496. Function ob_write_protection_get .....	361
Table 3-497. Function fmc_no_rtdec_config .....	362
Table 3-498. Function fmc_aes_iv_config .....	362
Table 3-499. Function fmc_flash_ecc_get .....	363
Table 3-500. Function fmc_no_rtdec_get.....	363
Table 3-501. Function fmc_aes_iv_get .....	364
Table 3-502. Function fmc_pid_get .....	364
Table 3-503. Function fmc_flag_get .....	365
Table 3-504. Function fmc_flag_clear .....	365
Table 3-505. Function fmc_interrupt_enable .....	366
Table 3-506. Function fmc_interrupt_disable.....	366
Table 3-507. Function fmc_interrupt_flag_get .....	367
Table 3-508. Function fmc_interrupt_flag_clear .....	367
Table 3-509. FWDGT Registers .....	368
Table 3-510. FWDGT firmware function .....	368
Table 3-511. Function fwdgt_write_ensable .....	368
Table 3-512. Function fwdgt_write_disable .....	369
Table 3-513. Function fwdgt_enable .....	369
Table 3-514. Function fwdgt_prescaler_value_config .....	370
Table 3-515. Function fwdgt_reload_value_config.....	370
Table 3-516. Function fwdgt_window_value_config .....	371
Table 3-517. Function fwdgt_counter_reload.....	371
Table 3-518. Function fwdgt_config.....	372
Table 3-519. Function fwdgt_flag_get.....	372
Table 3-520. GPIO Registers.....	373
Table 3-521. GPIO firmware function .....	374
Table 3-522. Function gpio_deinit .....	374
Table 3-523. Function gpio_mode_set.....	375

Table 3-524. Function gpio_output_options_set .....	376
Table 3-525. Function gpio_bit_set .....	377
Table 3-526. Function gpio_bit_reset .....	377
Table 3-527. Function gpio_bit_write.....	378
Table 3-528. Function gpio_port_write .....	378
Table 3-529. Function gpio_input_filter_set .....	379
Table 3-530. Function gpio_input_bit_get.....	380
Table 3-531. Function gpio_input_port_get.....	381
Table 3-532. Function gpio_output_bit_get .....	381
Table 3-533. Function gpio_output_port_get .....	382
Table 3-534. Function gpio_af_set .....	382
Table 3-535. Function gpio_pin_lock .....	383
Table 3-536. Function gpio_bit_toggle .....	384
Table 3-537. Function gpio_port_toggle .....	384
Table 3-538. HPDF Registers .....	385
Table 3-539. HPDF firmware function .....	386
Table 3-540. Structure hpdf_channel_parameter_struct.....	388
Table 3-541. Structure hpdf_filter_parameter_struct.....	389
Table 3-542. Structure hpdf_rc_parameter_struct.....	389
Table 3-543. Structure hpdf_ic_parameter_struct .....	389
Table 3-544. Enum hpdf_channel_enum .....	389
Table 3-545. Enum hpdf_filter_enum .....	390
Table 3-546. Enum hpdf_flag_enum.....	390
Table 3-547. Enum hpdf_interrput_flag_enum .....	392
Table 3-548. Enum hpdf_interrput_enum.....	393
Table 3-549. Function hpdf_deinit .....	393
Table 3-550. Function hpdf_channel_struct_para_init .....	394
Table 3-551. Function hpdf_filter_struct_para_init .....	394
Table 3-552. Function hpdf_rc_struct_para_init .....	395
Table 3-553. Function hpdf_ic_struct_para_init .....	396
Table 3-554. Function hpdf_enable .....	396
Table 3-555. Function hpdf_disable .....	397
Table 3-556. Function hpdf_channel_init.....	397
Table 3-557. Function hpdf_filter_init .....	398
Table 3-558. Function hpdf_rc_init.....	399
Table 3-559. Function hpdf_ic_init .....	400
Table 3-560. Function hpdf_clock_output_config .....	400
Table 3-561. Function hpdf_clock_output_source_config .....	401
Table 3-562. Function hpdf_clock_output_duty_mode_disable .....	402
Table 3-563. Function hpdf_clock_output_duty_mode_enable .....	402
Table 3-564. Function hpdf_clock_output_divider_config .....	403
Table 3-565. Function hpdf_channel_enable.....	403
Table 3-566. Function hpdf_channel_disable.....	404
Table 3-567. Function hpdf_spi_clock_source_config.....	404



Table 3-568. Function <code>hpdf_serial_interface_type_config</code> .....	405
Table 3-569. Function <code>hpdf_malfunction_monitor_disable</code> .....	405
Table 3-570. Function <code>hpdf_malfunction_monitor_enable</code> .....	406
Table 3-571. Function <code>hpdf_clock_loss_disable</code> .....	406
Table 3-572. Function <code>hpdf_clock_loss_enable</code> .....	407
Table 3-573. Function <code>hpdf_channel_pin_redirection_disable</code> .....	407
Table 3-574. Function <code>hpdf_channel_pin_redirection_enable</code> .....	408
Table 3-575. Function <code>hpdf_channel_multiplexer_config</code> .....	408
Table 3-576. Function <code>hpdf_data_pack_mode_config</code> .....	409
Table 3-577. Function <code>hpdf_data_right_bit_shift_config</code> .....	410
Table 3-578. Function <code>hpdf_calibration_offset_config</code> .....	410
Table 3-579. Function <code>hpdf_malfunction_break_signal_config</code> .....	411
Table 3-580. Function <code>hpdf_malfunction_counter_config</code> .....	412
Table 3-581. Function <code>hpdf_write_parallel_data_standard_mode</code> .....	412
Table 3-582. Function <code>hpdf_write_parallel_data_interleaved_mode</code> .....	413
Table 3-583. Function <code>hpdf_write_parallel_data_dual_mode</code> .....	413
Table 3-584. Function <code>hpdf_pulse_skip_update</code> .....	414
Table 3-585. Function <code>hpdf_pulse_skip_read</code> .....	414
Table 3-586. Function <code>hpdf_filter_enable</code> .....	415
Table 3-587. Function <code>hpdf_filter_disable</code> .....	415
Table 3-588. Function <code>hpdf_filter_config</code> .....	416
Table 3-589. Function <code>hpdf_integrator_oversample</code> .....	417
Table 3-590. Function <code>hpdf_threshold_monitor_filter_config</code> .....	417
Table 3-591. Function <code>hpdf_threshold_monitor_filter_read_data</code> .....	418
Table 3-592. Function <code>hpdf_threshold_monitor_fast_mode_disable</code> .....	418
Table 3-593. Function <code>hpdf_threshold_monitor_fast_mode_enable</code> .....	419
Table 3-594. Function <code>hpdf_threshold_monitor_channel</code> .....	419
Table 3-595. Function <code>hpdf_threshold_monitor_high_threshold</code> .....	420
Table 3-596. Function <code>hpdf_threshold_monitor_low_threshold</code> .....	421
Table 3-597. Function <code>hpdf_high_threshold_break_signal</code> .....	421
Table 3-598. Function <code>hpdf_low_threshold_break_signal</code> .....	422
Table 3-599. Function <code>hpdf_extremes_monitor_channel</code> .....	422
Table 3-600. Function <code>hpdf_extremes_monitor_maximum_get</code> .....	423
Table 3-601. Function <code>hpdf_extremes_monitor_minimum_get</code> .....	424
Table 3-602. Function <code>hpdf_conversion_time_get</code> .....	424
Table 3-603. Function <code>hpdf_rc_continuous_disable</code> .....	425
Table 3-604. Function <code>hpdf_rc_continuous_enable</code> .....	425
Table 3-605. Function <code>hpdf_rc_start_by_software</code> .....	426
Table 3-606. Function <code>hpdf_rc_syn_disable</code> .....	426
Table 3-607. Function <code>hpdf_rc_syn_enable</code> .....	427
Table 3-608. Function <code>hpdf_rc_dma_disable</code> .....	427
Table 3-609. Function <code>hpdf_rc_dma_enable</code> .....	428
Table 3-610. Function <code>hpdf_rc_channel_config</code> .....	428
Table 3-611. Function <code>hpdf_rc_fast_mode_disable</code> .....	429



Table 3-612. Function <code>hpdf_rc_fast_mode_enable</code> .....	429
Table 3-613. Function <code>hpdf_rc_data_get</code> .....	430
Table 3-614. Function <code>hpdf_rc_channel_get</code> .....	430
Table 3-615. Function <code>hpdf_ic_start_by_software</code> .....	431
Table 3-616. Function <code>hpdf_ic_syn_disable</code> .....	431
Table 3-617. Function <code>hpdf_ic_syn_enable</code> .....	432
Table 3-618. Function <code>hpdf_ic_dma_disable</code> .....	432
Table 3-619. Function <code>hpdf_ic_dma_enable</code> .....	433
Table 3-620. Function <code>hpdf_ic_scan_mode_disable</code> .....	433
Table 3-621. Function <code>hpdf_ic_scan_mode_enable</code> .....	434
Table 3-622. Function <code>hpdf_ic_trigger_signal_disable</code> .....	434
Table 3-623. Function <code>hpdf_ic_trigger_signal_config</code> .....	435
Table 3-624. Function <code>hpdf_ic_channel_config</code> .....	436
Table 3-625. Function <code>hpdf_ic_data_get</code> .....	436
Table 3-626. Function <code>hpdf_ic_channel_get</code> .....	437
Table 3-627. Function <code>hpdf_flag_get</code> .....	437
Table 3-628. Function <code>hpdf_flag_clear</code> .....	439
Table 3-629. Function <code>hpdf_interrupt_enable</code> .....	440
Table 3-630. Function <code>hpdf_interrupt_disable</code> .....	440
Table 3-631. Function <code>hpdf_interrupt_flag_get</code> .....	441
Table 3-632. Function <code>hpdf_interrupt_flag_clear</code> .....	442
Table 3-633. I2C Registers .....	443
Table 3-634. I2C firmware function.....	444
Table 3-635. Enum <code>i2c_interrupt_flag_enum</code> .....	446
Table 3-636. Function <code>i2c_deinit</code> .....	446
Table 3-637. Function <code>i2c_timing_config</code> .....	447
Table 3-638. Function <code>i2c_digital_noise_filter_config</code> .....	447
Table 3-639. Function <code>i2c_analog_noise_filter_enable</code> .....	448
Table 3-640. Function <code>i2c_analog_noise_filter_disable</code> .....	449
Table 3-641. Function <code>i2c_master_clock_config</code> .....	449
Table 3-642. Function <code>i2c_master_addressing</code> .....	450
Table 3-643. Function <code>i2c_address10_header_enable</code> .....	450
Table 3-644. Function <code>i2c_address10_header_disable</code> .....	451
Table 3-645. Function <code>i2c_address10_enable</code> .....	451
Table 3-646. Function <code>i2c_address10_disable</code> .....	452
Table 3-647. Function <code>i2c_automatic_end_enable</code> .....	452
Table 3-648. Function <code>i2c_automatic_end_disable</code> .....	453
Table 3-649. Function <code>i2c_slave_response_to_gcall_enable</code> .....	453
Table 3-650. Function <code>i2c_slave_response_to_gcall_disable</code> .....	454
Table 3-651. Function <code>i2c_stretch_scl_low_enable</code> .....	454
Table 3-652. Function <code>i2c_stretch_scl_low_disable</code> .....	455
Table 3-653. Function <code>i2c_address_config</code> .....	455
Table 3-654. Function <code>i2c_address_bit_compare_config</code> .....	456
Table 3-655. Function <code>i2c_address_disable</code> .....	457

Table 3-656. Function i2c_second_address_config.....	458
Table 3-657. Function i2c_second_address_disable .....	459
Table 3-658. Function i2c_receved_address_get .....	459
Table 3-659. Function i2c_slave_byte_control_enable.....	460
Table 3-660. Function i2c_slave_byte_control_disable.....	460
Table 3-661. Function i2c_nack_enable .....	461
Table 3-662. Function i2c_wakeup_from_deepsleep_enable.....	461
Table 3-663. Function i2c_wakeup_from_deepsleep_disable.....	462
Table 3-664. Function i2c_enable .....	462
Table 3-665. Function i2c_disable .....	463
Table 3-666. Function i2c_start_on_bus .....	463
Table 3-667. Function i2c_stop_on_bus.....	464
Table 3-668. Function i2c_data_transmit .....	464
Table 3-669. Function i2c_data_receive .....	465
Table 3-670. Function i2c_reload_enable.....	465
Table 3-671. Function i2c_reload_disable.....	466
Table 3-672. Function i2c_transfer_byte_number_config.....	466
Table 3-673. Function i2c_dma_enable .....	467
Table 3-674. Function i2c_dma_disable .....	467
Table 3-675. Function i2c_pec_transfer .....	468
Table 3-676. Function i2c_pec_enable.....	468
Table 3-677. Function i2c_pec_disable.....	469
Table 3-678. Function i2c_pec_value_get .....	469
Table 3-679. Function i2c_smbus_alert_enable.....	470
Table 3-680. Function i2c_smbus_alert_disable.....	470
Table 3-681. Function i2c_smbus_default_addr_enable .....	471
Table 3-682. Function i2c_smbus_default_addr_disable .....	471
Table 3-683. Function i2c_smbus_host_addr_enable .....	472
Table 3-684. Function i2c_smbus_host_addr_disable .....	472
Table 3-685. Function i2c_extented_clock_timeout_enable.....	473
Table 3-686. Function i2c_extented_clock_timeout_disable.....	473
Table 3-687. Function i2c_clock_timeout_enable.....	474
Table 3-688. Function i2c_clock_timeout_disable.....	474
Table 3-689. Function i2c_bus_timeout_b_config.....	475
Table 3-690. Function i2c_bus_timeout_a_config .....	476
Table 3-691. Function i2c_idle_clock_timeout_config.....	476
Table 3-692. Function i2c_flag_get.....	477
Table 3-693. Function i2c_flag_clear.....	478
Table 3-694. Function i2c_interrupt_enable.....	478
Table 3-695. Function i2c_interrupt_disable .....	479
Table 3-696. Function i2c_interrupt_flag_get.....	480
Table 3-697. Function i2c_interrupt_flag_clear .....	481
Table 3-698. LPDTS Registers .....	482
Table 3-699. LPDTS firmware function .....	482

Table3-700. Structure lpdts_parameter_struct .....	483
Table3-701. Function lpdts_deinit .....	483
Table3-702. Function lpdts_struct_para_init.....	484
Table3-703. Function lpdts_init.....	484
Table3-704. Function lpdts_enable .....	485
Table3-705. Function lpdts_disable .....	485
Table3-706. Function lpdts_soft_trigger_enable .....	486
Table3-707. Function lpdts_soft_trigger_disable .....	486
Table3-708. Function lpdts_high_threshold_set .....	487
Table3-709. Function lpdts_low_threshold_set.....	487
Table3-710. Function lpdts_ref_clock_source_config.....	488
Table3-711. Function lpdts_temperature_get.....	488
Table3-712. Function lpdts_flag_get.....	489
Table3-713. Function lpdts_interrupt_enable.....	489
Table3-714. Function lpdts_interrupt_disable.....	490
Table3-715. Function lpdts_interrupt_flag_get .....	490
Table3-716. Function lpdts_interrupt_flag_clear .....	491
Table 3-717. MDMA Registers .....	492
Table 3-718. MDMA firmware function .....	493
Table 3-719. Structure mdma_parameter_struct .....	494
Table 3-720. Structure mdma_multi_block_parameter_struct .....	494
Table 3-721. Structure mdma_link_node_parameter_struct.....	495
Table 3-722. Enum mdma_add_update_dir_enum .....	495
Table 3-723. Enum mdma_channel_enum .....	495
Table 3-724. Function mdma_deinit .....	496
Table 3-725. Function mdma_channel_deinit.....	496
Table 3-726. Function mdma_para_struct_init.....	497
Table 3-727. Function mdma_multi_block_para_struct_init.....	497
Table 3-728. Function mdma_link_node_para_struct_init .....	498
Table 3-729. Function mdma_init .....	498
Table 3-730. Function mdma_buffer_block_mode_config.....	500
Table 3-731. Function mdma_multi_block_mode_config .....	500
Table 3-732. Function mdma_node_create.....	501
Table 3-733. Function mdma_node_add .....	502
Table 3-734. Function mdma_node_delete .....	502
Table 3-735. Function mdma_destination_address_config .....	503
Table 3-736. Function mdma_source_address_config.....	504
Table 3-737. Function mdma_destination_bus_config.....	504
Table 3-738. Function mdma_source_bus_config .....	505
Table 3-739. Function mdma_priority_config.....	506
Table 3-740. Function mdma_endianness_config.....	506
Table 3-741. Function mdma_alignment_config.....	507
Table 3-742. Function mdma_source_burst_beats_config.....	508
Table 3-743. Function mdma_destination_burst_beats_config .....	509

Table 3-744. Function mdma_source_width_config .....	510
Table 3-745. Function mdma_destination_width_config .....	511
Table 3-746. Function mdma_source_increment_config .....	511
Table 3-747. Function mdma_destination_increment_config .....	512
Table 3-748. Function mdma_channel_bufferable_write_enable .....	513
Table 3-749. Function mdma_channel_bufferable_write_disable .....	514
Table 3-750. Function mdma_channel_software_request_enable .....	514
Table 3-751. Function mdma_channel_enable.....	515
Table 3-752. Function mdma_channel_disable.....	516
Table 3-753. Function mdma_transfer_error_direction_get .....	516
Table 3-754. Function mdma_transfer_error_address_get.....	517
Table 3-755. Function mdma_flag_get.....	517
Table 3-756. Function mdma_flag_clear .....	518
Table 3-757. Function mdma_interrupt_enable .....	519
Table 3-758. Function mdma_interrupt_disable .....	520
Table 3-759. Function mdma_interrupt_flag_get.....	520
Table 3-760. Function mdma_interrupt_flag_clear.....	521
Table 3-761. OSPI Registers .....	522
Table 3-762. OSPI firmware function.....	523
Table3-763. Structure ospi_parameter_struct.....	524
Table3-764. Structure ospi_regular_cmd_struct .....	525
Table3-765. Structure ospi_autopolling_struct.....	526
Table 3-766. Enum ospi_interrupt_flag_enum .....	526
Table3-767. Function ospi_deinit .....	527
Table3-768. Function ospi_struct_init .....	527
Table3-769. Function ospi_init.....	528
Table3-770. Function ospi_enable.....	528
Table3-771. Function ospi_disable .....	529
Table3-772. Function ospi_device_memory_type_config.....	529
Table3-773. Function ospi_device_memory_size_config .....	530
Table3-774. Function ospi_functional_mode_config.....	531
Table3-775. Function ospi_status_polling_config .....	531
Table3-776. Function ospi_status_mask_config .....	532
Table3-777. Function ospi_status_match_config.....	533
Table3-778. Function ospi_interval_cycle_config .....	533
Table3-779. Function ospi_fifo_level_config.....	534
Table3-780. Function ospi_chip_select_high_cycle_config.....	534
Table3-781. Function ospi_prescaler_config .....	535
Table3-782. Function ospi_dummy_cycles_config.....	535
Table3-783. Function ospi_delay_hold_cycle_config.....	536
Table3-784. Function ospi_sample_shift_config .....	537
Table3-785. Function ospi_data_length_config.....	537
Table3-786. Function ospi_instruction_config.....	538
Table3-787. Function ospi_address_config.....	539

Table3-788. Function <code>ospi_alternate_bytes_config</code> .....	540
Table3-789. Function <code>ospi_data_config</code> .....	541
Table3-790. Function <code>ospi_data_transmit</code> .....	542
Table3-791. Function <code>ospi_data_receive</code> .....	542
Table3-792. Function <code>ospi_dma_enable</code> .....	543
Table3-793. Function <code>ospi_dma_disable</code> .....	543
Table3-794. Function <code>ospi_wrap_size_config</code> .....	544
Table3-795. Function <code>ospi_wrap_instruction_config</code> .....	545
Table3-796. Function <code>ospi_wrap_address_config</code> .....	546
Table3-797. Function <code>ospi_wrap_alternate_bytes_config</code> .....	547
Table3-798. Function <code>ospi_wrap_data_config</code> .....	548
Table3-799. Function <code>ospi_wrap_dummy_cycles_config</code> .....	549
Table3-800. Function <code>ospi_wrap_delay_hold_cycle_config</code> .....	549
Table3-801. Function <code>ospi_wrap_sample_shift_config</code> .....	550
Table3-802. Function <code>ospi_write_instruction_config</code> .....	551
Table3-803. Function <code>ospi_write_address_config</code> .....	552
Table3-804. Function <code>ospi_write_alternate_bytes_config</code> .....	553
Table3-805. Function <code>ospi_write_data_config</code> .....	554
Table3-806. Function <code>ospi_write_dummy_cycles_config</code> .....	555
Table3-807. Function <code>ospi_write_dummy_cycles_config</code> .....	556
Table3-808. Function <code>ospi_transmit</code> .....	556
Table3-809. Function <code>ospi_receive</code> .....	557
Table3-810. Function <code>ospi_autopolling_mode</code> .....	557
Table3-811. Function <code>ospi_interrupt_enable</code> .....	558
Table3-812. Function <code>ospi_interrupt_disable</code> .....	559
Table3-813. Function <code>ospi_fifo_level_get</code> .....	559
Table3-814. Function <code>ospi_flag_get</code> .....	560
Table3-815. Function <code>ospi_flag_clear</code> .....	560
Table3-816. Function <code>ospi_interrupt_flag_get</code> .....	561
Table3-817. Function <code>ospi_interrupt_flag_clear</code> .....	561
Table 3-818. OSPIM Registers .....	562
Table 3-819. OSPIM firmware function .....	562
Table3-820. Function <code>ospim_deinit</code> .....	563
Table3-821. Function <code>ospim_port_sck_config</code> .....	563
Table3-822. Function <code>ospim_port_csn_config</code> .....	564
Table3-823. Function <code>ospim_port_io3_0_config</code> .....	564
Table3-824. Function <code>ospim_port_io3_0_source_select</code> .....	565
Table3-825. Function <code>ospim_port_io7_4_config</code> .....	566
Table3-826. Function <code>ospim_port_io7_4_source_select</code> .....	566
Table 3-827. NVIC Registers .....	567
Table 3-828. SysTick Registers .....	568
Table 3-829. MISC firmware function .....	569
Table 3-830. Structure <code>mpu_region_init_struct</code> .....	569
Table 3-831. Enum <code>IRQn_Type</code> .....	569

Table 3-832. Function nvic_priority_group_set .....	573
Table 3-833. Function nvic_irq_enable .....	574
Table 3-834. Function nvic_irq_disable .....	574
Table 3-835. Function nvic_vector_table_set .....	575
Table 3-836. Function system_lowpower_set .....	575
Table 3-837. Function system_lowpower_reset .....	576
Table 3-838. Function systick_clksource_set .....	576
Table 3-839. Function mpu_region_struct_para_init .....	577
Table 3-840. Function mpu_region_config .....	577
Table 3-841. Function mpu_region_enable .....	578
Table 3-842. PMU Registers .....	579
Table 3-843. PMU firmware function .....	579
Table 3-844. Function pmu_deinit .....	580
Table 3-845. Function pmu_lvd_select .....	581
Table 3-846. Function pmu_lvd_enable .....	581
Table 3-847. Function pmu_lvd_disable .....	582
Table 3-848. Function pmu_vavd_select .....	582
Table 3-849. Function pmu_vavd_enable .....	583
Table 3-850. Function pmu_vavd_disable .....	583
Table 3-851. Function pmu_cvd_enable .....	584
Table 3-852. Function pmu_vovd_disable .....	584
Table 3-853. Function pmu_ldo_output_select .....	585
Table 3-854. Function pmu_slido_output_select .....	585
Table 3-855. Function pmu_vbat_charging_select .....	586
Table 3-856. Function pmu_vbat_charging_enable .....	586
Table 3-857. Function pmu_vbat_charging_disable .....	587
Table 3-858. Function pmu_vbat_temp_moniter_enable .....	587
Table 3-859. Function pmu_vbat_temp_moniter_disable .....	588
Table 3-860. Function pmu_usb_regulator_enable .....	588
Table 3-861. Function pmu_usb_regulator_disable .....	589
Table 3-862. Function pmu_usb_voltage_detector_enable .....	589
Table 3-863. Function pmu_usb_voltage_detector_disable .....	590
Table 3-864. Function pmu_smps_ldo_supply_config .....	590
Table 3-865. Function pmu_to_sleepmode .....	591
Table 3-866. Function pmu_to_deepsleepmode .....	591
Table 3-867. Function pmu_to_standbymode .....	592
Table 3-868. Function pmu_wakeup_pin_enable .....	592
Table 3-869. Function pmu_wakeup_pin_disable .....	593
Table 3-870. Function pmu_backup_write_enable .....	593
Table 3-871. Function pmu_backup_write_disable .....	594
Table 3-872. Function pmu_backup_voltage_stabilizer_enable .....	594
Table 3-873. Function pmu_backup_voltage_stabilizer_disable .....	595
Table 3-874. Function pmu_enter_deepsleep_wait_time_config .....	595
Table 3-875. Function pmu_exit_deepsleep_wait_time_config .....	596

Table 3-876. Function pmu_flag_get.....	596
Table 3-877. Function pmu_flag_clear.....	597
Table 3-878. RAMECCMU Registers.....	598
Table 3-879. RAMECCMU firmware function .....	598
Table 3-880. Enum rameccmu_monitor_enum.....	599
Table 3-881. Function rameccmu_deinit .....	599
Table 3-882. Function rameccmu_monitor_failing_address_get.....	600
Table 3-883. Function rameccmu_monitor_failing_data_low_bits_get .....	601
Table 3-884. Function rameccmu_monitor_failing_data_high_bits_get .....	601
Table 3-885. Function rameccmu_monitor_failing_ecc_error_code_get.....	602
Table 3-886. Function rameccmu_global_interrupt_enable .....	603
Table 3-887. Function rameccmu_global_interrupt_disable .....	604
Table 3-888. Function rameccmu_monitor_interrupt_enable .....	605
Table 3-889. Function rameccmu_monitor_interrupt_disable .....	606
Table 3-890. Function rameccmu_monitor_flag_get.....	607
Table 3-891. Function rameccmu_monitor_flag_clear .....	608
Table 3-892. Function rameccmu_monitor_interrupt_flag_get .....	609
Table 3-893. Function rameccmu_monitor_interrupt_flag_clear .....	611
Table 3-894. RCU Registers .....	612
Table 3-895. RCU firmware function .....	613
Table 3-896. Enum rcu_periph_enum .....	615
Table 3-897. Enum rcu_periph_sleep_enum .....	618
Table 3-898. Enum rcu_periph_reset_enum.....	620
Table 3-899. Enum rcu_flag_enum.....	622
Table 3-900. Enum rcu_int_flag_enum .....	623
Table 3-901. Enum rcu_int_flag_clear_enum .....	624
Table 3-902. Enum rcu_int_enum.....	624
Table 3-903. Enum rcu_osci_type_enum .....	625
Table 3-904. Enum rcu_clock_freq_enum.....	625
Table 3-905. Enum usart_idx_enum.....	626
Table 3-906. Enum i2c_idx_enum.....	626
Table 3-907. Enum can_idx_enum.....	626
Table 3-908. Enum adc_idx_enum.....	626
Table 3-909. Enum usbhs_idx_enum .....	626
Table 3-910. Enum pll_idx_enum.....	627
Table 3-911. Enum spi_idx_enum.....	627
Table 3-912. Function rcu_deinit .....	627
Table 3-913. Function rcu_periph_clock_enable .....	628
Table 3-914. Function rcu_periph_clock_disable .....	628
Table 3-915. Function rcu_periph_clock_sleep_enable .....	629
Table 3-916. Function rcu_periph_clock_sleep_disable .....	629
Table 3-917. Function rcu_periph_reset_enable.....	629
Table 3-918. Function rcu_periph_reset_disable .....	630
Table 3-919. Function rcu_bkp_reset_enable .....	630



Table 3-920. Function rcu_bkp_reset_disable .....	631
Table 3-921. Function rcu_system_clock_source_config.....	631
Table 3-922. Function rcu_system_clock_source_get .....	632
Table 3-923. Function rcu_ahb_clock_config .....	633
Table 3-924. Function rcu_apb1_clock_config .....	633
Table 3-925. Function rcu_apb2_clock_config .....	634
Table 3-926. Function rcu_apb3_clock_config .....	634
Table 3-927. Function rcu_apb4_clock_config .....	635
Table 3-928. Function rcu_ckout0_config.....	635
Table 3-929. Function rcu_ckout1_config.....	636
Table 3-930. Function rcu_pll_input_output_clock_range_config.....	637
Table 3-931. Function rcu_pll_fractional_config .....	638
Table 3-932. Function rcu_pll_fractional_latch_enable.....	638
Table 3-933. Function rcu_pll_fractional_latch_disable.....	639
Table 3-934. Function rcu_pll_source_config .....	639
Table 3-935. Function rcu_pll0_config .....	640
Table 3-936. Function rcu_pll1_config .....	641
Table 3-937. Function rcu_pll2_config .....	641
Table 3-938. Function rcu_pll_clock_output_enable .....	642
Table 3-939. Function rcu_pll_clock_output_disable .....	643
Table 3-940. Function rcu_pllusb0_config.....	644
Table 3-941. Function rcu_pllusb1_config.....	645
Table 3-942. Function rcu_rtc_clock_config .....	645
Table 3-943. Function rcu_rtc_div_config .....	646
Table 3-944. Function rcu_ck48m_clock_config .....	647
Table 3-945. Function rcu_pll48m_clock_config .....	647
Table 3-946. Function rcu_irc64mdiv_clock_config .....	648
Table 3-947. Function rcu_irc64mdiv_freq_get.....	648
Table 3-948. Function rcu_timer_clock_prescaler_config.....	649
Table 3-949. Function rcu_spi_clock_config.....	649
Table 3-950. Function rcu_deepsleep_wakeup_sys_clock_config .....	650
Table 3-951. Function rcu_usart_clock_config.....	651
Table 3-952. Function rcu_i2c_clock_config.....	651
Table 3-953. Function rcu_can_clock_config.....	652
Table 3-954. Function rcu_adc_clock_config.....	653
Table 3-955. Function rcu_exmc_clock_config .....	653
Table 3-956. Function rcu_hpdf_clock_config.....	654
Table 3-957. Function rcu_per_clock_config .....	655
Table 3-958. Function rcu_usbhs_pll1qpsc_config .....	655
Table 3-959. Function rcu_usb48m_clock_config.....	656
Table 3-960. Function rcu_usbhs_clock_config .....	656
Table 3-961. Function rcu_usbhs_clock_selection_enable .....	657
Table 3-962. Function rcu_usbhs_clock_selection_disable.....	658
Table 3-963. Function rcu_lxtal_drive_capability_config.....	658



Table 3-964. Function rcu_osc_i_stab_wait .....	659
Table 3-965. Function rcu_osc_i_on .....	659
Table 3-966. Function rcu_osc_i_off .....	660
Table 3-967. Function rcu_osc_i_bypass_mode_enable .....	660
Table 3-968. Function rcu_osc_i_bypass_mode_disable .....	661
Table 3-969. Function rcu_irc64m_adjust_value_set.....	661
Table 3-970. Function rcu_lpirc4m_adjust_value_set .....	661
Table 3-971. Function rcu_hxtal_clock_monitor_enable .....	662
Table 3-972. Function rcu_hxtal_clock_monitor_disable .....	662
Table 3-973. Function rcu_lxtal_clock_monitor_enable.....	663
Table 3-974. Function rcu_lxtal_clock_monitor_disable.....	663
Table 3-975. Function rcu_clock_freq_get.....	664
Table 3-976. Function rcu_flag_get.....	664
Table 3-977. Function rcu_all_reset_flag_clear .....	665
Table 3-978. Function rcu_interrupt_enable.....	665
Table 3-979. Function rcu_interrupt_disable.....	666
Table 3-980. Function rcu_interrupt_flag_get .....	666
Table 3-981. Function rcu_interrupt_flag_clear .....	667
Table 3-982. RTC Registers .....	668
Table 3-983. RTC firmware function.....	669
Table 3-984. Structure rtc_parameter_struct.....	670
Table 3-985. Structure rtc_alarm_struct.....	670
Table 3-986. Structure rtc_timestamp_struct.....	671
Table 3-987. Structure rtc_tamper_struct .....	671
Table 3-988. Function rtc_deinit .....	671
Table 3-989. Function rtc_init.....	672
Table 3-990. Function rtc_init_mode_enter .....	673
Table 3-991. Function rtc_init_mode_exit.....	673
Table 3-992. Function rtc_register_sync_wait .....	674
Table 3-993. Function rtc_current_time_get.....	674
Table 3-994. Function rtc_subsecond_get.....	675
Table 3-995. Function rtc_alarm_config.....	675
Table 3-996. Function rtc_alarm_subsecond_config.....	676
Table 3-997. Function rtc_alarm_enable .....	677
Table 3-998. Function rtc_alarm_disable .....	678
Table 3-999. Function rtc_alarm_get.....	678
Table 3-1000. Function rtc_alarm_subsecond_get .....	679
Table 3-1001. Function rtc_timestamp_enable .....	679
Table 3-1002. Function rtc_timestamp_disable .....	680
Table 3-1003. Function rtc_timestamp_internalevent_config .....	680
Table 3-1004. Function rtc_timestamp_get.....	681
Table 3-1005. Function rtc_timestamp_subsecond_get .....	681
Table 3-1006. Function rtc_tamper_enable.....	682
Table 3-1007. Function rtc_tamper_disable.....	682

Table 3-1008. Function rtc_output_pin_select.....	683
Table 3-1009. Function rtc_alarm_output_config.....	683
Table 3-1010. Function rtc_calibration_output_config.....	684
Table 3-1011. Function rtc_hour_adjust.....	685
Table 3-1012. Function rtc_second_adjust.....	685
Table 3-1013. Function rtc_bypass_shadow_enable .....	686
Table 3-1014. Function rtc_bypass_shadow_disable .....	686
Table 3-1015. Function rtc_refclock_detection_enable.....	687
Table 3-1016. Function rtc_refclock_detection_disable.....	687
Table 3-1017. Function rtc_wakeup_enable .....	688
Table 3-1018. Function rtc_wakeup_disable .....	688
Table 3-1019. Function rtc_wakeup_clock_set .....	689
Table 3-1020. Function rtc_wakeup_timer_set .....	689
Table 3-1021. Function rtc_wakeup_timer_get .....	690
Table 3-1022. Function rtc_smooth_calibration_config.....	690
Table 3-1023. Function rtc_interrupt_enable.....	691
Table 3-1024. Function rtc_interrupt_disable.....	692
Table 3-1025. Function rtc_flag_get.....	693
Table 3-1026. Function rtc_flag_clear.....	693
Table 3-1027. SPI/I2S Registers .....	694
Table 3-1028. SPI/I2S firmware function.....	695
Table 3-1029. spi_parameter_struct.....	697
Table 3-1030. Function spi_i2s_deinit .....	697
Table 3-1031. Function spi_struct_para_init .....	698
Table 3-1032. Function spi_init.....	698
Table 3-1033. Function spi_enable.....	699
Table 3-1034. Function spi_disable.....	699
Table 3-1035. Function i2s_init .....	700
Table 3-1036. Function i2s_psc_config .....	701
Table 3-1037. Function i2s_enable .....	702
Table 3-1038. Function i2s_disable .....	703
Table 3-1039. Function spi_io_config .....	703
Table 3-1040. Function spi_nss_idleness_delay_set.....	704
Table 3-1041. Function spi_data_frame_delay_set .....	704
Table 3-1042. Function spi_master_receive_clock_delay_set.....	705
Table 3-1043. Function spi_slave_receive_clock_delay_set.....	706
Table 3-1044. Function spi_master_receive_clock_delay_clear.....	706
Table 3-1045. Function spi_slave_receive_clock_delay_clear .....	707
Table 3-1046. Function spi_nss_output_control .....	707
Table 3-1047. Function spi_nss_polarity_set.....	708
Table 3-1048. Function spi_nss_output_enable .....	708
Table 3-1049. Function spi_nss_output_disable .....	709
Table 3-1050. Function spi_nss_internal_high .....	709
Table 3-1051. Function spi_nss_internal_low .....	710

Table 3-1052. Function spi_dma_enable .....	710
Table 3-1053. Function spi_dma_disable .....	711
Table 3-1054. Function spi_i2s_data_frame_size_config .....	712
Table 3-1055. Function spi_i2s_data_transmit .....	712
Table 3-1056. Function spi_i2s_data_receive .....	713
Table 3-1057. Function spi_bidirectional_transfer_config .....	713
Table 3-1058. Function spi_master_transfer_start.....	714
Table 3-1059. Function spi_current_data_num_config .....	715
Table 3-1060. Function spi_reload_data_num_config .....	715
Table 3-1061. Function spi_crc_polynomial_set.....	716
Table 3-1062. Function spi_crc_polynomial_get .....	716
Table 3-1063. Function spi_crc_length_config .....	717
Table 3-1064. Function spi_crc_on .....	717
Table 3-1065. Function spi_crc_off .....	718
Table 3-1066. Function spi_crc_get .....	718
Table 3-1067. Function spi_crc_full_size_enable.....	719
Table 3-1068. Function spi_crc_full_size_disable .....	719
Table 3-1069. Function spi_tcrc_init_pattern .....	720
Table 3-1070. Function spi_rcrc_init_pattern .....	720
Table 3-1071. Function spi_ti_mode_enable .....	721
Table 3-1072. Function spi_ti_mode_disable .....	722
Table 3-1073. Function spi_quad_enable.....	722
Table 3-1074. Function spi_quad_disable.....	723
Table 3-1075. Function spi_quad_write_enable.....	723
Table 3-1076. Function spi_quad_read_enable.....	724
Table 3-1077. Function spi_quad_io23_output_enable .....	724
Table 3-1078. Function spi_quad_io23_output_disable .....	725
Table 3-1079. Function spi_underrun_operation.....	725
Table 3-1080. Function spi_underrun_config.....	726
Table 3-1081. Function spi_underrun_data_config.....	726
Table 3-1082. Function spi_suspend_mode_config .....	727
Table 3-1083. Function spi_suspend_request .....	728
Table 3-1084. Function spi_related_ios_af_enable .....	728
Table 3-1085. Function spi_related_ios_af_disable .....	729
Table 3-1086. Function spi_af_gpio_control .....	729
Table 3-1087. Function spi_i2s_interrupt_enable.....	730
Table 3-1088. Function spi_i2s_interrupt_disable.....	731
Table 3-1089. Function spi_i2s_interrupt_flag_get .....	731
Table 3-1090. Function spi_i2s_flag_get .....	733
Table 3-1091. Function spi_i2s_flag_clear.....	734
Table 3-1092. Function spi_i2s_rxfifo_plevel_get .....	734
Table 3-1093. Function spi_i2s_remain_data_num_get .....	735
Table 3-1094. Function spi_fifo_threshold_level_set.....	736
Table 3-1095. Function spi_word_access_enable .....	736

Table 3-1096. Function spi_word_access_disable .....	737
Table 3-1097. Function spi_byte_access_enable .....	737
Table 3-1098. Function spi_byte_access_disable .....	738
Table 3-1099. SYSCFG Registers .....	738
Table 3-1100. SYSCFG firmware function .....	739
Table 3-1101. Enum timer_channel_input_enum .....	740
Table 3-1102. Function syscfg_deinit .....	745
Table 3-1103. Function syscfg_i2c_fast_mode_plus_enable .....	746
Table 3-1104. Function syscfg_i2c_fast_mode_plus_disable .....	746
Table 3-1105. Function syscfg_analog_switch_enable .....	747
Table 3-1106. Function syscfg_analog_switch_disable .....	748
Table 3-1107. Function syscfg_enet_phy_interface_config .....	748
Table 3-1108. Function syscfg_exti_line_config .....	749
Table 3-1109. Function syscfg_lockup_enable .....	750
Table 3-1110. Function syscfg_lockup_disable .....	751
Table 3-1111. Function syscfg_timer_input_source_select .....	751
Table 3-1112. Function syscfg_compensation_config .....	752
Table 3-1113. Function syscfg_io_low_voltage_speed_optimization_enable .....	753
Table 3-1114. Function syscfg_io_low_voltage_speed_optimization_disable .....	753
Table 3-1115. Function syscfg_pnmos_compensation_code_set .....	753
Table 3-1116. Function syscfg_secure_sram_size_set .....	754
Table 3-1117. Function syscfg_secure_sram_size_get .....	755
Table 3-1118. Function syscfg_bootmode_get .....	755
Table 3-1119. Function syscfg_tcm_wait_state_enable .....	756
Table 3-1120. Function syscfg_tcm_wait_state_disable .....	757
Table 3-1121. Function syscfg_fpu_interrupt_enable .....	757
Table 3-1122. Function syscfg_fpu_interrupt_disable .....	758
Table 3-1123. Function syscfg_compensation_flag_get .....	759
Table 3-1124. Function syscfg_cpu_cache_status_get .....	759
Table 3-1125. Function syscfg_brownout_reset_threshold_level_get .....	760
Table 3-1126. TIMERx Registers .....	761
Table 3-1127. TIMERx firmware function .....	762
Table 3-1128. Structure timer_parameter_struct .....	765
Table 3-1129. Structure timer_break_parameter_struct .....	766
Table 3-1130. Structure timer_oc_parameter_struct .....	767
Table 3-1131. Structure timer_omc_parameter_struct .....	767
Table 3-1132. Structure timer_ic_parameter_struct .....	767
Table 3-1133. Structure timer_free_complementary_parameter_struct .....	768
Table 3-1134. Function timer_deinit .....	768
Table 3-1135. Function timer_struct_para_init .....	768
Table 3-1136. Function timer_init .....	769
Table 3-1137. Function timer_enable .....	770
Table 3-1138. Function timer_disable .....	770
Table 3-1139. Function timer_auto_reload_shadow_enable .....	771

Table 3-1140. Function timer_auto_reload_shadow_disable .....	771
Table 3-1141. Function timer_update_event_enable.....	772
Table 3-1142. Function timer_update_event_disable.....	772
Table 3-1143. Function timer_counter_alignment .....	773
Table 3-1144. Function timer_counter_up_direction .....	774
Table 3-1145. Function timer_counter_down_direction .....	774
Table 3-1146. Function timer_prescaler_config.....	775
Table 3-1147. Function timer_repetition_value_config.....	775
Table 3-1148. Function timer_runtime_repetition_value_read .....	776
Table 3-1149. Function timer_autoreload_value_config .....	777
Table 3-1150. Function timer_autoreload_value_read .....	777
Table 3-1151. Function timer_counter_value_config .....	778
Table 3-1152. Function timer_counter_read .....	779
Table 3-1153. Function timer_prescaler_read .....	779
Table 3-1154. Function timer_single_pulse_mode_config .....	780
Table 3-1155. Function timer_delayable_single_pulse_mode_config .....	780
Table 3-1156. Function timer_update_source_config.....	782
Table 3-1157. Function timer_dma_enable .....	782
Table 3-1158. Function timer_dma_disable .....	783
Table 3-1159. Function timer_channel_dma_request_source_select.....	784
Table 3-1160. Function timer_dma_transfer_config.....	785
Table 3-1161. Function timer_event_software_generate.....	787
Table 3-1162. Function timer_break_struct_para_init .....	788
Table 3-1163. Function timer_break_config.....	789
Table 3-1164. Function timer_break_enable .....	790
Table 3-1165. Function timer_break_disable .....	791
Table 3-1166. Function timer_automatic_output_enable .....	791
Table 3-1167. Function timer_automatic_output_disable .....	792
Table 3-1168. Function timer_primary_output_config .....	792
Table 3-1169. Function timer_channel_control_shadow_config .....	793
Table 3-1170. Function timer_channel_control_shadow_update_config.....	794
Table 3-1171. Function timer_channel_output_struct_para_init .....	795
Table 3-1172. Function timer_channel_output_config .....	795
Table 3-1173. Function timer_channel_output_mode_config .....	796
Table 3-1174. Function timer_channel_output_pulse_value_config.....	797
Table 3-1175. Function timer_channel_output_shadow_config .....	798
Table 3-1176. Function timer_channel_output_clear_config .....	799
Table 3-1177. Function timer_channel_output_polarity_config.....	800
Table 3-1178. Function timer_channel_complementary_output_polarity_config.....	801
Table 3-1179. Function timer_channel_output_state_config .....	802
Table 3-1180. Function timer_channel_complementary_output_state_config .....	803
Table 3-1181. Function timer_channel_input_struct_para_init .....	804
Table 3-1182. Function timer_input_capture_config.....	805
Table 3-1183. Function timer_channel_input_capture_prescaler_config .....	806

Table 3-1184. Function timer_channel_capture_value_register_read .....	806
Table 3-1185. Function timer_input_pwm_capture_config .....	807
Table 3-1186. Function timer_hall_mode_config .....	808
Table 3-1187. Function timer_multi_mode_channel_output_parameter_struct_init .....	809
Table 3-1188. Function timer_multi_mode_channel_output_config .....	809
Table 3-1189. Function timer_multi_mode_channel_mode_config .....	810
Table 3-1190. Function timer_input_trigger_source_select .....	811
Table 3-1191. Function timer_master_output0_trigger_source_select .....	813
Table 3-1192. Function timer_master_output1_trigger_source_select .....	814
Table 3-1193. Function timer_slave_mode_select .....	815
Table 3-1194. Function timer_master_slave_mode_config .....	816
Table 3-1195. Function timer_external_trigger_config .....	817
Table 3-1196. Function timer_quadrature_decoder_mode_config .....	817
Table 3-1197. Function timer_non_quadrature_decoder_mode_config .....	819
Table 3-1198. Function timer_internal_clock_config .....	819
Table 3-1199. Function timer_internal_trigger_as_external_clock_config .....	820
Table 3-1200. Function timer_external_trigger_as_external_clock_config .....	821
Table 3-1201. Function timer_external_clock_mode0_config .....	822
Table 3-1202. Function timer_external_clock_mode1_config .....	823
Table 3-1203. Function timer_external_clock_mode1_disable .....	824
Table 3-1204. Function timer_write_chxval_register_config .....	825
Table 3-1205. Function timer_output_value_selection_config .....	825
Table 3-1206. Function timer_commutation_control_shadow_register_config .....	826
Table 3-1207. Function timer_output_match_pulse_select .....	827
Table 3-1208. Function timer_channel_composite_pwm_mode_config .....	828
Table 3-1209. Function timer_channel_composite_pwm_mode_output_pulse_value_config .....	828
Table 3-1210. Function timer_channel_additional_compare_value_config .....	829
Table 3-1211. Function timer_channel_additional_output_shadow_config .....	830
Table 3-1212. Function timer_channel_additional_compare_value_read .....	831
Table 3-1213. Function timer_break_external_source_config .....	832
Table 3-1214. Function timer_break_external_polarity_config .....	832
Table 3-1215. Function timer_break_lock_config .....	833
Table 3-1216. Function timer_break_lock_release_config .....	834
Table 3-1217. Function timer_channel_break_control_config .....	835
Table 3-1218. Function timer_channel_dead_time_config .....	836
Table 3-1219. Function timer_free_complementary_struct_para_init .....	836
Table 3-1220. Function timer_channel_free_complementary_config .....	837
Table 3-1221. Function timer_watchdog_value_config .....	838
Table 3-1222. Function timer_watchdog_value_read .....	838
Table 3-1223. Function timer_decoder_disconnection_detection_config .....	839
Table 3-1224. Function timer_decoder_jump_detection_config .....	840
Table 3-1225. Function timer_upif_backup_config .....	840
Table 3-1226. Function timer_upifbu_bit_get .....	841
Table 3-1227. Function timer_flag_get .....	841

Table 3-1228. Function timer_flag_clear .....	843
Table 3-1229. Function timer_interrupt_enable .....	844
Table 3-1230. Function timer_interrupt_disable .....	845
Table 3-1231. Function timer_interrupt_flag_get.....	846
Table 3-1232. Function timer_interrupt_flag_clear.....	848
Table 3-1233. TMU Registers .....	850
Table 3-1234. TMU firmware function .....	850
Table 3-1235. Structure tmu_parameter_struct .....	850
Table 3-1236. Function tmu_deinit .....	851
Table 3-1237. Function tmu_struct_para_init.....	851
Table 3-1238. Function tmu_init.....	852
Table 3-1239. Function tmu_read_interrupt_enable .....	853
Table 3-1240. Function tmu_read_interrupt_disable .....	853
Table 3-1241. Function tmu_dma_read_enable .....	854
Table 3-1242. Function tmu_dma_read_disable .....	854
Table 3-1243. Function tmu_dma_write_enable .....	854
Table 3-1244. Function tmu_dma_write_disable .....	855
Table 3-1245. Function tmu_one_q31_write .....	855
Table 3-1246. Function tmu_two_q31_write .....	856
Table 3-1247. Function tmu_two_q15_write .....	857
Table 3-1248. Function tmu_one_q31_read .....	857
Table 3-1249. Function tmu_two_q31_read .....	858
Table 3-1250. Function tmu_two_q15_read .....	858
Table 3-1251. TRIGSEL Registers.....	859
Table 3-1252. TRIGSEL firmware function .....	860
Table 3-1253. Enum trigsel_source_enum.....	860
Table 3-1254. Enum trigsel_periph_enum .....	864
Table 3-1255. Function trigsel_deinit.....	865
Table 3-1256. Function trigsel_init .....	866
Table 3-1257. Function trigsel_trigger_source_get .....	866
Table 3-1258. Function trigsel_register_lock_set.....	867
Table 3-1259. Function trigsel_register_lock_get .....	868
Table 3-1260. TRNG Registers .....	868
Table 3-1261. TRNG firmware function.....	868
Table 3-1262. Enum trng_inmod_enum.....	869
Table 3-1263. Enum trng_outmod_enum .....	869
Table 3-1264. Enum trng_modsel_enum .....	870
Table 3-1265. Enum trng_flag_enum .....	870
Table 3-1266. Enum trng_int_flag_enum.....	870
Table 3-1267. Function trng_deinit.....	870
Table 3-1268. Function trng_enable.....	871
Table 3-1269. Function trng_disable.....	871
Table 3-1270. Function trng_lock .....	872
Table 3-1271. Function trng_mode_config .....	872



Table 3-1272. Function <code>trng_postprocessing_enable</code> .....	873
Table 3-1273. Function <code>trng_postprocessing_disable</code> .....	874
Table 3-1274. Function <code>trng_conditioning_enable</code> .....	874
Table 3-1275. Function <code>trng_conditioning_disable</code> .....	875
Table 3-1276. Function <code>trng_disable</code> .....	876
Table 3-1277. Function <code>trng_conditioning_output_bitwidth</code> .....	877
Table 3-1278. Function <code>trng_replace_test_enable</code> .....	877
Table 3-1279. Function <code>trng_replace_test_disable</code> .....	878
Table 3-1280. Function <code>trng_hash_init_enable</code> .....	879
Table 3-1281. Function <code>trng_hash_init_disable</code> .....	879
Table 3-1282. Function <code>trng_powermode_config</code> .....	880
Table 3-1283. Function <code>trng_disable</code> .....	881
Table 3-1284. Function <code>trng_clockerror_detection_enable</code> .....	881
Table 3-1285. Function <code>trng_clockerror_detection_disable</code> .....	882
Table 3-1286. Function <code>trng_get_true_random_data</code> .....	883
Table 3-1287. Function <code>trng_conditioning_reset_enable</code> .....	883
Table 3-1288. Function <code>trng_conditioning_reset_disable</code> .....	884
Table 3-1289. Function <code>trng_conditioning_algo_config</code> .....	885
Table 3-1290. Function <code>trng_health_tests_config</code> .....	886
Table 3-1291. Function <code>trng_flag_get</code> .....	886
Table 3-1292. Function <code>trng_interrupt_enable</code> .....	887
Table 3-1293. Function <code>trng_interrupt_disable</code> .....	887
Table 3-1294. Function <code>trng_interrupt_flag_get</code> .....	888
Table 3-1295. Function <code>trng_interrupt_flag_clear</code> .....	888
Table 3-1296. USART Registers .....	889
Table 3-1297. USART firmware function .....	889
Table 3-1298. Enum <code>usart_flag_enum</code> .....	892
Table 3-1299. Enum <code>usart_interrupt_flag_enum</code> .....	893
Table 3-1300. Enum <code>usart_interrupt_enum</code> .....	893
Table 3-1301. Enum <code>usart_invert_enum</code> .....	894
Table 3-1302. Function <code>usart_deinit</code> .....	894
Table 3-1303. Function <code>usart_baudrate_set</code> .....	895
Table 3-1304. Function <code>usart_parity_config</code> .....	895
Table 3-1305. Function <code>usart_word_length_set</code> .....	896
Table 3-1306. Function <code>usart_stop_bit_set</code> .....	897
Table 3-1307. Function <code>usart_enable</code> .....	897
Table 3-1308. Function <code>usart_disable</code> .....	898
Table 3-1309. Function <code>usart_transmit_config</code> .....	898
Table 3-1310. Function <code>usart_receive_config</code> .....	899
Table 3-1311. Function <code>usart_data_first_config</code> .....	900
Table 3-1312. Function <code>usart_invert_config</code> .....	900
Table 3-1313. Function <code>usart_overrun_enable</code> .....	901
Table 3-1314. Function <code>usart_overrun_disable</code> .....	901
Table 3-1315. Function <code>usart_oversample_config</code> .....	902



Table 3-1316. Function usart_sample_bit_config.....	902
Table 3-1317. Function usart_receiver_timeout_enable .....	903
Table 3-1318. Function usart_receiver_timeout_disable .....	903
Table 3-1319. Function usart_receiver_timeout_threshold_config.....	904
Table 3-1320. Function usart_data_transmit .....	905
Table 3-1321. Function usart_data_receive .....	905
Table 3-1322. Function usart_command_enable .....	906
Table 3-1323. Function usart_address_0_match_mode_enable.....	906
Table 3-1324. Function usart_address_0_match_mode_disable.....	907
Table 3-1325. Function usart_address_1_match_mode_enable.....	907
Table 3-1326. Function usart_address_1_match_mode_disable.....	908
Table 3-1327. Function usart_address_0_config.....	908
Table 3-1328. Function usart_address_1_config.....	909
Table 3-1329. Function usart_address_0_detection_mode_config.....	910
Table 3-1330. Function usart_address_1_detection_mode_config.....	910
Table 3-1331. Function usart_mute_mode_enable.....	911
Table 3-1332. Function usart_mute_mode_disable .....	911
Table 3-1333. Function usart_mute_mode_wakeup_config .....	912
Table 3-1334. Function usart_lin_mode_enable .....	913
Table 3-1335. Function usart_lin_mode_disable .....	913
Table 3-1336. Function usart_lin_break_detection_length_config.....	914
Table 3-1337. Function usart_halfduplex_enable.....	914
Table 3-1338. Function usart_halfduplex_disable.....	915
Table 3-1339. Function usart_clock_enable .....	915
Table 3-1340. Function usart_clock_disable .....	916
Table 3-1341. Function usart_synchronous_clock_config .....	916
Table 3-1342. Function usart_guard_time_config .....	917
Table 3-1343. Function usart_smartcard_mode_enable .....	918
Table 3-1344. Function usart_smartcard_mode_disable .....	918
Table 3-1345. Function usart_smartcard_mode_nack_enable.....	919
Table 3-1346. Function usart_smartcard_mode_nack_disable .....	919
Table 3-1347. Function usart_smartcard_mode_early_nack_enable.....	920
Table 3-1348. Function usart_smartcard_mode_early_nack_disable.....	920
Table 3-1349. Function usart_smartcard_autoretry_config.....	921
Table 3-1350. Function usart_block_length_config .....	921
Table 3-1351. Function usart_irda_mode_enable.....	922
Table 3-1352. Function usart_irda_mode_disable.....	922
Table 3-1353. Function usart_prescaler_config.....	923
Table 3-1354. Function usart_irda_lowpower_config .....	923
Table 3-1355. Function usart_hardware_flow_rts_config .....	924
Table 3-1356. Function usart_hardware_flow_cts_config .....	925
Table 3-1357. Function usart_hardware_flow_coherence_config .....	925
Table 3-1358. Function usart_rs485_driver_enable .....	926
Table 3-1359. Function usart_rs485_driver_disable .....	926

Table 3-1360. Function usart_driver_asserttime_config .....	927
Table 3-1361. Function usart_driver_deasserttime_config .....	928
Table 3-1362. Function usart_depolarity_config .....	928
Table 3-1363. Function usart_dma_receive_config .....	929
Table 3-1364. Function usart_dma_transmit_config .....	929
Table 3-1365. Function usart_reception_error_dma_disable .....	930
Table 3-1366. Function usart_reception_error_dma_enable .....	931
Table 3-1367. Function usart_wakeup_enable .....	931
Table 3-1368. Function usart_wakeup_disable .....	932
Table 3-1369. Function usart_wakeup_mode_config .....	932
Table 3-1370. Function usart_fifo_enable .....	933
Table 3-1371. Function usart_fifo_disable .....	933
Table 3-1372. Function usart_transmit_fifo_threshold_config .....	934
Table 3-1373. Function usart_receive_fifo_threshold_config .....	935
Table 3-1374. Function usart_receive_fifo_counter_number .....	936
Table 3-1375. Function usart_flag_get .....	936
Table 3-1376. Function usart_flag_clear .....	937
Table 3-1377. Function usart_interrupt_enable .....	938
Table 3-1378. Function usart_interrupt_disable .....	938
Table 3-1379. Function usart_interrupt_flag_get .....	939
Table 3-1380. Function usart_interrupt_flag_clear .....	939
Table 3-1381. VREF Registers .....	941
Table 3-1382. VREF firmware function .....	941
Table 3-1383. Function vref_deinit .....	941
Table 3-1384. Function vref_enable .....	942
Table 3-1385. Function vref_disable .....	942
Table 3-1386. Function vref_high_impedance_mode_enable .....	943
Table 3-1387. Function vref_high_impedance_mode_disable .....	943
Table 3-1388. Function vref_status_get .....	944
Table 3-1389. Function vref_voltage_select .....	944
Table 3-1390. Function vref_calib_value_set .....	945
Table 3-1391. Function vref_calib_value_get .....	946
Table 3-1392. WWDGT Registers .....	946
Table 3-1393. WWDGT firmware function .....	946
Table 3-1394. Function wwdgt_deinit .....	947
Table 3-1395. Function wwdgt_enable .....	947
Table 3-1396. Function wwdgt_counter_update .....	948
Table 3-1397. Function wwdgt_config .....	948
Table 3-1398. Function wwdgt_interrupt_enable .....	949
Table 3-1399. Function wwdgt_flag_get .....	949
Table 3-1400. Function wwdgt_flag_clear .....	950
Table 3-1401. ESC_INTC Registers .....	951
Table 3-1402. ESC_INTC firmware function .....	951
Table 3-1403. Enum intc_enable_enum .....	951

Table 3-1404. Enum intc_disable_enum.....	952
Table 3-1405. Enum intc_get_flag_enum .....	952
Table 3-1406. Enum intc_clear_flag_enum .....	952
Table 3-1407. Function intc_deas_config .....	953
Table 3-1408. Function intc_deasc_config .....	953
Table 3-1409. Function intc_irq_config .....	954
Table 3-1410. Function intc_interrupt_enable.....	955
Table 3-1411. Function intc_interrupt_disable.....	955
Table 3-1412. Function intc_interrupt_flag_get .....	956
Table 3-1413. Function intc_interrupt_flag_clear .....	956
Table 3-1414. Function intc_deasstat_get .....	957
Table 3-1415. Function intc_irqstat_get.....	957
Table 3-1416. ESC_OSPI Registers .....	958
Table 3-1417. ESC_OSPI firmware function.....	959
Table 3-1418. Function ospi_hw_init .....	959
Table 3-1419. Function ospi_enable_qspi_mode .....	960
Table 3-1420. Function ospi_enable_ospi_mode .....	960
Table 3-1421. Function ospi_reset_spi_mode.....	961
Table 3-1422. Function ospi_write.....	962
Table 3-1423. Function ospi_read .....	963
Table 3-1424. Function OSPIWriteRegUsingCSR .....	964
Table 3-1425. Function OSPIReadRegUsingCSR .....	965
Table 3-1426. Function OSPIReadRegister .....	967
Table 3-1427. Function OSPIWriteRegister.....	967
Table 3-1428. ESC_PHY Registers .....	968
Table 3-1429. ESC_PHY firmware function .....	968
Table 3-1430. Function esc_phy_read .....	969
Table 3-1431. Function esc_phy_write .....	969
Table 3-1432. Function esc_mmd_read.....	970
Table 3-1433. Function esc_mmd_write.....	971
Table 3-1434. ESC_PMU Registers.....	971
Table 3-1435. ESC_PMU firmware function .....	971
Table 3-1436. Enum pmu_esc_fun_enum .....	972
Table 3-1437. Function pmu_esc_power_mang_mode_config.....	972
Table 3-1438. Function pmu_esc_wake_up_mode_config .....	973
Table 3-1439. Function pmu_esc_led_wm_config .....	974
Table 3-1440. Function pmu_esc_led_inact_stat_config .....	974
Table 3-1441. Function pmu_esc_ready_stat_get .....	975
Table 3-1442. Function pmu_esc_edwol_stat_get .....	975
Table 3-1443. Function pmu_esc_edwol_stat_clear .....	976
Table 3-1444. Function pmu_esc_fun_config .....	976
Table 3-1445. Function pmu_esc_byte_test .....	977
Table 3-1446. Function pmu_esc_sleep_mode_enable.....	977
Table 3-1447. Function pmu_esc_sleep_mode_disable.....	978

Table 3-1448. ESC_SYSCFG Registers.....	978
Table 3-1449. ESC_SYSCFG firmware function .....	979
Table 3-1450. Function syscfg_get_chip_id .....	979
Table 3-1451. Function syscfg_get_chip_version .....	980
Table 3-1452. Function syscfg_efuse_read .....	980
Table 3-1453. ESC_TIMER Registers .....	981
Table 3-1454. ESC_TIMER firmware function.....	981
Table 3-1455. Function esc_timer_enable .....	982
Table 3-1456. Function esc_timer_disable .....	982
Table 3-1457. Function esc_timer_autoreload_value_config.....	983
Table 3-1458. Function esc_timer_counter_read .....	983
Table 3-1459. Function esc_frc_counter_read .....	984
Table 4-1. Revision history .....	985

## 1. Introduction

This manual introduces firmware library of GD32H75E 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 GD32H75E 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
CMP	Comparator
CRC	CRC calculation unit
CTC	Clock trim controller

Peripherals	Descriptions
DAC	Digital-to-analog converter
DBG	Debug
DMA	Direct memory access controller
DMAMUX	DMA request multiplexer
EDOUT	Encoder Divided-Output controller
EFUSE	Electronic fuse
EXMC	External memory controller
EXTI	Interrupt/event controller
FAC	Filter arithmetic accelerator
FMC	Flash memory controller
FWDGT	Free watchdog timer
GPIO	General-purpose I/Os
HPDF	High-Performance Digital Filter
I2C	Inter-integrated circuit interface
LPDTS	Low power digital temperature sensor
MDMA	Master direct memory access controller
OSPI	Octal-SPI interface
OSPIIM	OSPI I/O manager
MISC	Nested Vectored Interrupt Controller
PMU	Power management unit
RAMECCMU	RAM ECC monitor unit
RCU	Reset and clock unit
RTC	Reset and clock unit
SPI/I2S	Secure digital input/output interface
SYSCFG	System configuration
TIMER	TIMER
TMU	Trigonometric Math Unit
TRIGSEL	Trigger selection controller
TRNG	True random number generator
USART	Universal synchronous / asynchronous receiver / transmitter
VREF	VREF
WWDGT	Window watchdog timer
ESC	EtherCAT Slave Controller

### 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 “gd32h75e\_”, such as: gd32h75e\_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.

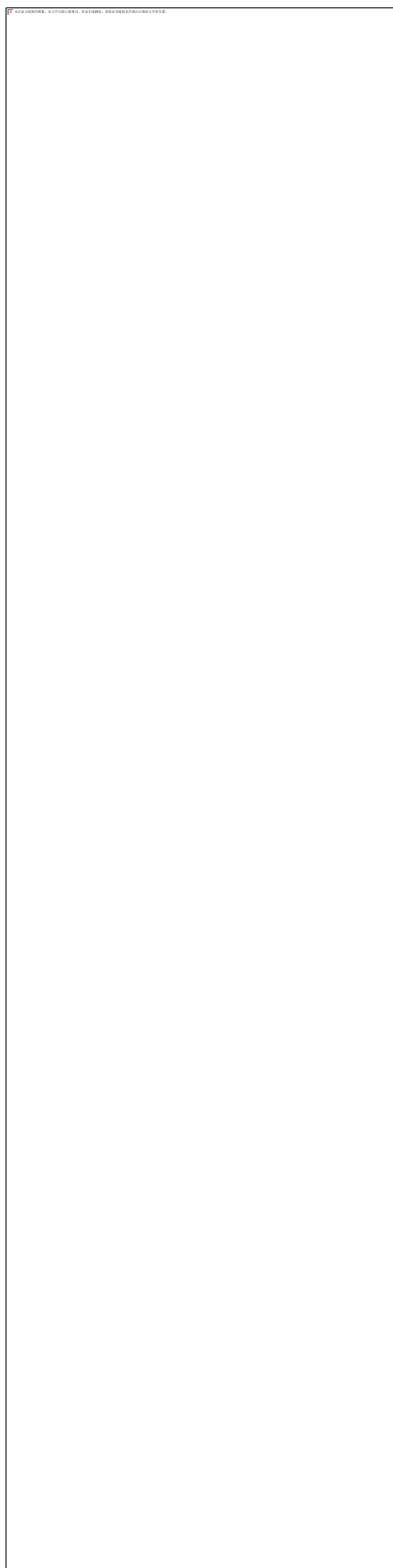
## **2. Firmware Library Overview**

### **2.1. File Structure of Firmware Library**

GD32H75E\_Firmware\_Library, the file structure is shown as below:



**Figure 2-1. File structure of firmware library of GD32H75E**



### 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;
- gd32h75e\_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);
- gd32h75e\_it.c: the source file include all the interrupt service routines (if no interrupt is used, then all the function bodies are empty);
- gd32h75e\_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® M7 kernel support files, the startup file based on the Cortex® M7 kernel processor, the global header file of GD32H75E and system configuration file;
- GD32H75E\_standard\_peripheral subfolder:
  - Include subfolder includes all the header files of firmware library, users need not modify this folder;
  - Source subfolder includes all the source files of firmware library, users need not modify this folder;

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

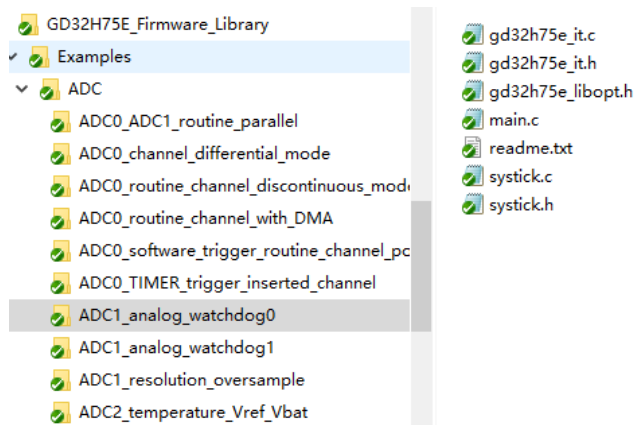
### 2.1.3. Template Folder

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

#### Select files

Open "Examples" folder, select the module to be tested, such as ADC, open "ADC" folder, select an example of ADC, such as "ADC1\_analog\_watchdog0", shown as below:

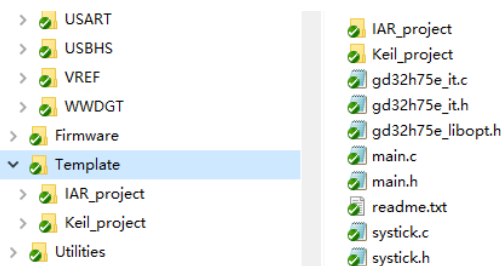
**Figure 2-2. Select peripheral example files**



## Copy files

Open "Template" folder, keep the folders of " IAR\_project" and " Keil\_project", and delete the other files, then copy all the files in "ADC1\_analog\_watchdog0" folder to the "Template" subfolder, shown as below:

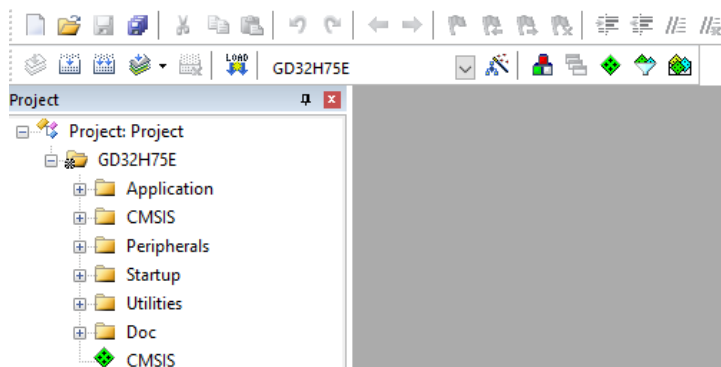
**Figure 2-3. Copy the peripheral example files**



## Open a project

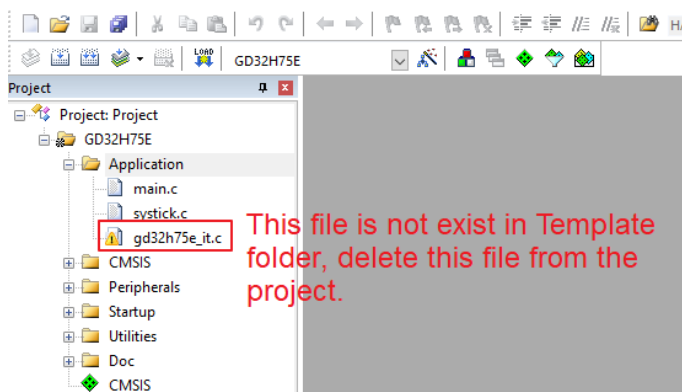
GD provides project in Keil and IAR, users can open project in different IDEs according to their need, such as "Keil\_project", open \Template\Keil\_project\Project.uvprojx, shown as below:

**Figure 2-4. Open the project file**



Because different module and different functions adopt different files, users should add or delete the files in project according to the copied files, shown as below:

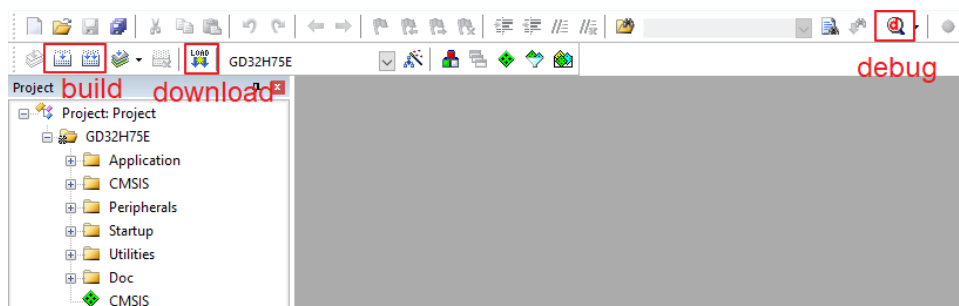
Figure 2-5. Configure project files



## Compile-Debug-Download

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

Figure 2-6. Compile-debug-download



### 2.1.4. Utilities Folder

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

- gd32h75ey\_eval.h is related header file of the evaluation board about running the firmware examples;
- gd32h75ey\_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
gd32h75e_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.
gd32h75e_it.h	Header file, including all the prototypes of interrupt service routines.
gd32h75e_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.
gd32h75e_xxx.h	The header file of peripheral xxx, including functions about peripheral xxx, and the variables used for functions.
gd32h75e_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 functin
<b>Input parameter{in}</b>	
<b>Input parameter name</b>	Description
xxxx	Description of input parameters
<b>Output parameter{out}</b>	
<b>Output parameter name</b>	Description
xxxx	Description of output parameters
<b>Return value</b>	
<b>Return value type</b>	The range of return value

### 3.2. ADC

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

#### 3.2.1. Descriptions of Peripheral registers

ADC registers are listed in the table shown as below:

**Table 3-2. ADC Registers**

<b>Registers</b>	<b>Descriptions</b>
ADC_STAT	ADC status register
ADC_CTL0	ADC control register 0
ADC_CTL1	ADC control register 1
ADC_IOFFx	ADC inserted channel data offset register x(x=0..3)
ADC_WDHT0	ADC watchdog high threshold register 0
ADC_WDLT0	ADC watchdog low threshold register 0
ADC_RSQx	ADC routine sequence register x(x=0..8)
ADC_ISQx	ADC inserted sequence register x(x=0..2)

Registers	Descriptions
ADC_IDATAx	ADC inserted data register x(x=0..3)
ADC_RDATA	ADC routine data register
ADC_OVSAMPCTL	ADC oversampling control register
ADC_WD1SR	ADC watchdog 1 channel selection register
ADC_WD2SR	ADC watchdog 2 channel selection register
ADC_WDHT1	ADC watchdog high threshold register 1
ADC_WDLT1	ADC watchdog low threshold register 1
ADC_WDHT2	ADC watchdog high threshold register 2
ADC_WDLT2	ADC watchdog low threshold register 2
ADC_DIFCTL	ADC differential mode control register
ADC_SSTAT	ADC summary status register
ADC_SYNCCTL	ADC sync control register
ADC_SYNCDATA0	ADC sync routine data register 0
ADC_SYNCDATA1	ADC sync routine data register 1

### 3.2.2. Descriptions of Peripheral functions

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

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

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



Function name	Function description
adc_routine_channel_config	configure ADC routine channel
adc_inserted_channel_config	configure ADC inserted channel
adc_inserted_channel_offset_config	configure ADC inserted channel offset
adc_channel_differential_mode_config	configure differential mode for channel
adc_external_trigger_config	enable ADC external trigger
adc_software_trigger_enable	enable ADC software trigger
adc_end_of_conversion_config	configure end of conversion mode
adc_routine_data_read	read ADC routine sequence data register
adc_inserted_data_read	read ADC inserted sequence data register
adc_watchdog0_single_channel_enable	configure ADC analog watchdog 0 single channel
adc_watchdog0_group_channel_enable	configure ADC analog watchdog 0 group channel
adc_watchdog0_disable	disable ADC analog watchdog 0
adc_watchdog1_channel_config	configure ADC analog watchdog 1 channel
adc_watchdog2_channel_config	configure ADC analog watchdog 2 channel
adc_watchdog1_disable	disable ADC analog watchdog 1
adc_watchdog2_disable	disable ADC analog watchdog 2
adc_watchdog0_threshold_config	configure ADC analog watchdog 0 threshold
adc_watchdog1_threshold_config	configure ADC analog watchdog 1 threshold
adc_watchdog2_threshold_config	configure ADC analog watchdog 2 threshold
adc_oversample_mode_config	configure ADC oversample mode
adc_oversample_mode_enable	enable ADC oversample mode
adc_oversample_mode_disable	disable ADC oversample mode
adc_flag_get	get the ADC flag bits
adc_flag_clear	clear the ADC flag bits
adc_interrupt_enable	enable ADC interrupt
adc_interrupt_disable	disable ADC interrupt
adc_interrupt_flag_get	get the ADC interrupt bits
adc_interrupt_flag_clear	clear the ADC flag
adc_sync_mode_config	configure the ADC sync mode
adc_sync_delay_config	configure the delay between 2 sampling phases in ADC sync modes
adc_sync_dma_config	configure ADC sync DMA mode selection
adc_sync_dma_request_after_last_enable	configure ADC sync DMA engine issues requests according to the SYNCDMA bits
adc_sync_dma_request_after_last_disable	configure ADC sync DMA engine is disabled after the end of transfer signal from DMA controller is detected
adc_sync_master_adc_routine_data0_read	read ADC sync master adc routine data register 0
adc_sync_slave_adc_routine_data0_read	read ADC sync slave adc routine data register 0

Function name	Function description
ead	
adc_sync_routine_data1_read	read ADC sync routine data register 1

## adc\_deinit

The description of adc\_deinit is shown as below:

**Table 3-4. Function adc\_deinit**

Function name	adc_deinit
Function prototype	void adc_deinit(uint32_t adc_periph);
Function descriptions	reset ADC
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

## adc\_clock\_config

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

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

Function name	adc_clock_config
Function prototype	void adc_clock_config(uint32_t adc_periph, uint32_t prescaler);
Function descriptions	configure the ADC clock for all the ADCs
Precondition	-
The called functions	-
Input parameter{in}	
adc_periph	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
Input parameter{in}	
prescaler	configure ADCs prescaler ratio
ADC_CLK_SYNC_HCLK_DIV2	ADC sync clock mode HCLK div2
ADC_CLK_SYNC_HCLK_DIV4	ADC sync clock mode HCLK div4

ADC_CLK_SYNC_HCL K_DIV6	ADC sync clock mode HCLK div6
ADC_CLK_SYNC_HCL K_DIV8	ADC sync clock mode HCLK div8
ADC_CLK_SYNC_HCL K_DIV10	ADC sync clock mode HCLK div10
ADC_CLK_SYNC_HCL K_DIV12	ADC sync clock mode HCLK div12
ADC_CLK_SYNC_HCL K_DIV14	ADC sync clock mode HCLK div14
ADC_CLK_SYNC_HCL K_DIV16	ADC sync clock mode HCLK div16
ADC_CLK_ASYNC_DI V1	ADC async clock mode div1
ADC_CLK_ASYNC_DI V2	ADC async clock mode div2
ADC_CLK_ASYNC_DI V4	ADC async clock mode div4
ADC_CLK_ASYNC_DI V6	ADC async clock mode div6
ADC_CLK_ASYNC_DI V8	ADC async clock mode div8
ADC_CLK_ASYNC_DI V10	ADC async clock mode div10
ADC_CLK_ASYNC_DI V12	ADC async clock mode div12
ADC_CLK_ASYNC_DI V16	ADC async clock mode div16
ADC_CLK_ASYNC_DI V32	ADC async clock mode div32
ADC_CLK_ASYNC_DI V64	ADC async clock mode div64
ADC_CLK_ASYNC_DI V128	ADC async clock mode div128
ADC_CLK_ASYNC_DI V256	ADC async clock mode div256
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### adc\_special\_function\_config

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

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

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

Example:

```
/* enable ADC0 scan mode */
```

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

### adc\_data\_alignment\_config

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

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

<b>Function name</b>	adc_data_alignment_config
<b>Function prototype</b>	void adc_data_alignment_config(uint32_t adc_periph, uint32_t data_alignment);
<b>Function descriptions</b>	configure ADCx data alignment

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>data_alignment</b>	data alignment select
<i>ADC_DATAALIGN_RIGHT</i>	LSB alignment
<i>ADC_DATAALIGN_LEFT</i>	MSB alignment
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

## adc\_enable

The description of adc\_enable is shown as below:

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

<b>Function name</b>	adc_enable
<b>Function prototype</b>	void adc_enable(uint32_t adc_periph);
<b>Function descriptions</b>	enable ADC interface
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2)</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 3-9. Function adc\_disable**

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

Example:

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

## adc\_calibration\_mode\_config

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

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

<b>Function name</b>	adc_calibration_mode_config
<b>Function prototype</b>	void adc_calibration_mode_config(uint32_t adc_periph, uint32_t clb_mode);
<b>Function descriptions</b>	configure ADC calibration mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>clb_mode</b>	calibration mode
<i>ADC_CALIBRATION_OFFSET_MISMATCH</i>	ADC calibration offset and mismatch mode
<i>ADC_CALIBRATION_OFFSET</i>	ADC calibration offset mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

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

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

### adc\_calibration\_number

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

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

<b>Function name</b>	adc_calibration_number
<b>Function prototype</b>	void adc_calibration_number(uint32_t adc_periph, uint32_t clb_num);
<b>Function descriptions</b>	configure ADC calibration number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>clb_num</b>	calibration number
ADC_CALIBRATION_NUM1	calibrate once
ADC_CALIBRATION_NUM2	calibrate twice
ADC_CALIBRATION_NUM4	calibrate 4 times
ADC_CALIBRATION_NUM8	calibrate 8 times
ADC_CALIBRATION_NUM16	calibrate 16 times
ADC_CALIBRATION_NUM32	calibrate 32 times
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 calibration number */
```

```
adc_calibration_number(ADC0, ADC_CALIBRATION_NUM1);
```



## adc\_calibration\_enable

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

**Table 3-12. Function adc\_calibration\_enable**

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

Example:

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

```
adc_calibration_enable(ADC0);
```

## adc\_resolution\_config

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

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

<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
ADCx(x=0, 1, 2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>resolution</b>	ADC resolution
ADC_RESOLUTION_14B	14-bit ADC resolution, for ADC0/ADC1
ADC_RESOLUTION_12B	12-bit ADC resolution, for all ADCs
ADC_RESOLUTION_10B	10-bit ADC resolution, for all ADCs
ADC_RESOLUTION_8B	8-bit ADC resolution, for all ADCs

<i>B</i>	
<i>ADC_RESOLUTION_6</i>	6-bit ADC resolution, only for ADC2
<i>B</i>	
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### adc\_internal\_channel\_config

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

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

<b>Function name</b>	adc_internal_channel_config
<b>Function prototype</b>	void adc_internal_channel_config(uint32_t internal_channel, ControlStatus newvalue);
<b>Function descriptions</b>	enable or disable ADC internal channels
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>internal_channel</b>	the internal channels
<i>ADC_CHANNEL_INTERRNAL_TEMPSENSOR</i>	temperature sensor channel
<i>ADC_CHANNEL_INTERRNAL_VREFINT</i>	vrefint channel
<i>ADC_CHANNEL_INTERRNAL_VBAT</i>	vbat channel
<i>ADC_CHANNEL_INTERRNAL_HP_TEMPSENS OR</i>	high-precision temperature sensor channel
<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 ADC temperature sensor channel */
```

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

### adc\_dma\_mode\_enable

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

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

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

Example:

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

```
adc_dma_mode_enable(ADC0);
```

### adc\_dma\_mode\_disable

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

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

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

Example:

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

```
adc_dma_mode_disable(ADC0);
```

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

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

**Table 3-17. Function adc\_dma\_request\_after\_last\_enable**

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

Example:

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

```
adc_dma_request_after_last_enable(ADC0);
```

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

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

**Table 3-18. Function adc\_dma\_request\_after\_last\_disable**

<b>Function name</b>	adc_dma_request_after_last_disable
<b>Function prototype</b>	void adc_dma_request_after_last_disable(uint32_t adc_periph);
<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
ADCx(x=0,1,2)	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

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

```
adc_dma_request_after_last_disable (ADC0);
```

### adc\_hpdf\_mode\_enable

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

**Table 3-19. Function adc\_hpdf\_mode\_enable**

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

Example:

```
/* enable ADC0 hpdf mode */
```

```
adc_hpdf_mode_enable(ADC0);
```

### adc\_hpdf\_mode\_disable

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

**Table 3-20. Function adc\_hpdf\_mode\_disable**

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

Return value	
-	-

Example:

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

### adc\_discontinuous\_mode\_config

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

**Table 3-21. Function adc\_discontinuous\_mode\_config**

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

Example:

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

### adc\_channel\_length\_config

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

Table 3-22. Function `adc_channel_length_config`

Function name	<code>adc_channel_length_config</code>
Function prototype	<code>void adc_channel_length_config(uint32_t adc_periph, uint8_t adc_sequence, uint32_t length);</code>
Function descriptions	configure the length of routine sequence or inserted sequence
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_periph</code>	ADC peripheral
<code>ADCx(x=0,1,2)</code>	ADC peripheral selection
Input parameter{in}	
<code>adc_sequence</code>	select the sequence
<code>ADC_ROUTINE_CHANNEL</code>	routine sequence
<code>ADC_INSERTED_CHANNEL</code>	inserted sequence
Input parameter{in}	
<code>length</code>	the length of the channel, routine channel 1-16, inserted channel 1-4
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

### `adc_routine_channel_config`

The description of `adc_routine_channel_config` is shown as below:

Table 3-23. Function `adc_routine_channel_config`

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



<b>adc_channel</b>	the selected ADC channel
<i>ADC_CHANNEL_x</i> (x=0..20)	ADC Channelx
<b>Input parameter{in}</b>	
<b>sample_time</b>	the sample time value x, x = 0..809 for ADC0/ADC1, 0..638 for ADC2. Eg.10'dx: For ADC0/1 is (x+3.5) cycles, For ADC2 is (x+2.5) cycles
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### adc\_inserted\_channel\_config

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

**Table 3-24. Function adc\_inserted\_channel\_config**

<b>Function name</b>	adc_inserted_channel_config
<b>Function prototype</b>	void adc_inserted_channel_config(uint32_t adc_periph, uint8_t rank, uint8_t adc_channel, uint32_t sample_time);
<b>Function descriptions</b>	configure ADC inserted channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>rank</b>	the inserted sequencer rank, this parameter must be between 0 to 3
<b>Input parameter{in}</b>	
<b>adc_channel</b>	the selected ADC channel
<i>ADC_CHANNEL_x</i> (x=0..20)	ADC Channelx
<b>Input parameter{in}</b>	
<b>sample_time</b>	the sample time value x, x = 0..809 for ADC0/ADC1, 0..638 for ADC2. Eg. 10'dx: For ADC0/1 is (x+3.5) cycles, For ADC2 is (x+2.5) cycles
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 inserted channel */
adc_inserted_channel_config(ADC0, 1, ADC_CHANNEL_0, 10);
```

### adc\_inserted\_channel\_offset\_config

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

**Table 3-25. Function adc\_inserted\_channel\_offset\_config**

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

Example:

```
/* configure ADC0 inserted channel offset */
adc_inserted_channel_offset_config(ADC0, ADC_INSERTED_CHANNEL_0, 100);
```

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

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

**Table 3-26. Function adc\_channel\_differential\_mode\_config**

<b>Function name</b>	adc_channel_differential_mode_config
<b>Function prototype</b>	void adc_channel_differential_mode_config(uint32_t adc_periph, uint32_t adc_channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure differential mode for ADC channel
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Input parameter{in}	
<b>adc_channel</b>	the channel use differential mode
<i>ADC_DIFFERENTIAL_MODE_CHANNEL_x</i> ( <i>x=0..21</i> ), <i>ADC_DIFFERENTIAL_MODE_CHANNEL_ALL</i>	ADC channel for differential mode
Input parameter{in}	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

### adc\_external\_trigger\_config

The description of `adc_external_trigger_config` is shown as below:

**Table 3-27. Function `adc_external_trigger_config`**

<b>Function name</b>	<code>adc_external_trigger_config</code>
<b>Function prototype</b>	<code>void adc_external_trigger_config(uint32_t adc_periph, uint8_t adc_sequence, uint32_t trigger_mode)</code>
<b>Function descriptions</b>	configure ADC external trigger
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
Input parameter{in}	
<b>adc_sequence</b>	select the sequence
<i>ADC_ROUTINE_CHANNEL</i>	routine sequence
<i>ADC_INSERTED_CHANNEL</i>	inserted sequence

Input parameter{in}	
<b>trigger_mode</b>	external trigger mode
<i>EXTERNAL_TRIGGER_DISABLE</i>	external trigger disable
<i>EXTERNAL_TRIGGER_RISING</i>	rising edge of external trigger
<i>EXTERNAL_TRIGGER_FALLING</i>	falling edge of external trigger
<i>EXTERNAL_TRIGGER_RISING_FALLING</i>	rising and falling edge of external trigger
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

### adc\_software\_trigger\_enable

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

**Table 3-28. Function adc\_software\_trigger\_enable**

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

Example:

```
/* enable ADC0 routine sequence software trigger */
adc_software_trigger_enable(ADC0, ADC_ROUTINE_CHANNEL);
```

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

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

**Table 3-29. Function adc\_end\_of\_conversion\_config**

<b>Function name</b>	adc_end_of_conversion_config
<b>Function prototype</b>	void adc_end_of_conversion_config(uint32_t adc_periph, uint32_t end_selection);
<b>Function descriptions</b>	configure end of conversion mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0, 1, 2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>end_selection</b>	end of conversion mode
ADC_EOC_SET_SEQUENCE	only at the end of a sequence of routine conversions, the EOC bit is set. Overflow detection is disabled unless DMA=1
ADC_EOC_SET_CONVERSION	at the end of each routine conversion, the EOC bit is set. Overflow is detected automatically
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC0 end of conversion mode */
adc_end_of_conversion_config(ADC0, ADC_EOC_SET_SEQUENCE);
```

### adc\_routine\_data\_read

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

**Table 3-30. Function adc\_routine\_data\_read**

<b>Function name</b>	adc_routine_data_read
<b>Function prototype</b>	uint32_t adc_routine_data_read(uint32_t adc_periph);
<b>Function descriptions</b>	read ADC routine sequence data register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0, 1, 2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	ADC conversion value

Example:

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

```
uint32_t adc_value = 0;
```

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

### adc\_inserted\_data\_read

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

**Table 3-31. Function adc\_inserted\_data\_read**

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

Example:

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

```
uint32_t adc_value = 0;
```

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

### adc\_watchdog0\_single\_channel\_enable

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

Table 3-32. Function `adc_watchdog0_single_channel_enable`

Function name	<code>adc_watchdog0_single_channel_enable</code>
Function prototype	<code>void adc_watchdog0_single_channel_enable(uint32_t adc_periph, uint8_t adc_channel);</code>
Function descriptions	configure ADC analog watchdog 0 single channel
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_periph</code>	ADC peripheral
<code>ADCx(x=0,1,2)</code>	ADC peripheral selection
Input parameter{in}	
<code>adc_channel</code>	the selected ADC channel
<code>ADC_CHANNEL_x</code> ( $x=0..20$ )	ADC channelx
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

### `adc_watchdog0_group_channel_enable`

The description of `adc_watchdog0_group_channel_enable` is shown as below:

Table 3-33. Function `adc_watchdog0_group_channel_enable`

Function name	<code>adc_watchdog0_group_channel_enable</code>
Function prototype	<code>void adc_watchdog0_group_channel_enable(uint32_t adc_periph, uint8_t adc_sequence);</code>
Function descriptions	configure ADC analog watchdog 0 group channel
Precondition	-
The called functions	-
Input parameter{in}	
<code>adc_periph</code>	ADC peripheral
<code>ADCx(x=0,1,2)</code>	ADC peripheral selection
Input parameter{in}	
<code>adc_sequence</code>	the sequence use analog watchdog
<code>ADC_ROUTINE_CHANNEL</code>	routine sequence
<code>ADC_INSERTED_CHANNEL</code>	inserted sequence
<code>ADC_ROUTINE_CHANNEL</code>	both routine and inserted sequence

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

Example:

```
/* configure ADC0 analog watchdog 0 group channel */
adc_watchdog0_group_channel_enable(ADC0, ADC_ROUTINE_CHANNEL);
```

### adc\_watchdog0\_disable

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

**Table 3-34. Function adc\_watchdog0\_disable**

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

Example:

```
/* disable ADC0 analog watchdog 0 */
adc_watchdog0_disable(ADC0);
```

### adc\_watchdog1\_channel\_config

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

**Table 3-35. Function adc\_watchdog1\_channel\_config**

Function name	adc_watchdog1_channel_config
Function prototype	void adc_watchdog1_channel_config(uint32_t adc_periph, uint32_t selection_channel, ControlStatus newvalue);
Function descriptions	configure ADC analog watchdog 1 channel
Precondition	-
The called functions	-
Input parameter{in}	



<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>selection_channel</b>	the channel use analog watchdog 1
<i>ADC_AWD1_2_SELECTION_CHANNEL_x</i> ( <i>x=0..20</i> ), <i>ADC_AWD1_2_SELECTION_CHANNEL_ALL</i>	ADC channel analog watchdog 1/2 selection
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### adc\_watchdog2\_channel\_config

The description of `adc_watchdog2_channel_config` is shown as below:

**Table 3-36. Function `adc_watchdog2_channel_config`**

<b>Function name</b>	<code>adc_watchdog2_channel_config</code>
<b>Function prototype</b>	<code>void adc_watchdog2_channel_config(uint32_t adc_periph, uint32_t selection_channel, ControlStatus newvalue);</code>
<b>Function descriptions</b>	configure ADC analog watchdog 2 channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>selection_channel</b>	the channel use analog watchdog 2
<i>ADC_AWD1_2_SELECTION_CHANNEL_x</i> ( <i>x=0..20</i> ), <i>ADC_AWD1_2_SELECTION_CHANNEL_ALL</i>	ADC channel analog watchdog 1/2 selection

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

Example:

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

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

### adc\_watchdog1\_disable

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

**Table 3-37. Function adc\_watchdog1\_disable**

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

Example:

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

```
adc_watchdog1_disable(ADC0);
```

### adc\_watchdog2\_disable

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

**Table 3-38. Function adc\_watchdog2\_disable**

<b>Function name</b>	adc_watchdog2_disable
<b>Function prototype</b>	void adc_watchdog2_disable(uint32_t adc_periph);
<b>Function descriptions</b>	disable ADC analog watchdog 2

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

Example:

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

```
adc_watchdog2_disable(ADC0);
```

### adc\_watchdog0\_threshold\_config

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

**Table 3-39. Function adc\_watchdog0\_threshold\_config**

<b>Function name</b>	adc_watchdog0_threshold_config
<b>Function prototype</b>	void adc_watchdog0_threshold_config(uint32_t adc_periph , uint32_t low_threshold , uint32_t high_threshold);
<b>Function descriptions</b>	configure ADC analog watchdog 0 threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<b>ADCx(x=0,1,2)</b>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>low_threshold</b>	analog watchdog 0 low threshold, 0..0xFFFF for ADC0/ADC1, 0..0xFFF for ADC2
<b>Input parameter{in}</b>	
<b>high_threshold</b>	analog watchdog 0 high threshold, 0..0xFFFF for ADC0/ADC1, 0..0xFFF for ADC2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

## adc\_watchdog1\_threshold\_config

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

**Table 3-40. Function adc\_watchdog1\_threshold\_config**

<b>Function name</b>	adc_watchdog1_threshold_config
<b>Function prototype</b>	void adc_watchdog1_threshold_config(uint32_t adc_periph , uint32_t low_threshold , uint32_t high_threshold);
<b>Function descriptions</b>	configure ADC analog watchdog 1 threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>low_threshold</b>	analog watchdog 1 low threshold, 0..0xFFFFFFFF for ADC0/ADC1, 0..0xFF for ADC2
<b>Input parameter{in}</b>	
<b>high_threshold</b>	analog watchdog 1 high threshold, 0..0xFFFFFFFF for ADC0/ADC1, 0..0xFF for ADC2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC2 analog watchdog 1 threshold */
```

```
adc_watchdog1_threshold_config(ADC2, 0x40, 0xA0);
```

## adc\_watchdog2\_threshold\_config

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

**Table 3-41. Function adc\_watchdog2\_threshold\_config**

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

<b>low_threshold</b>	analog watchdog 2 low threshold, 0..0xFFFFFF for ADC0/ADC1, 0..0xFF for ADC2
<b>Input parameter{in}</b>	
<b>high_threshold</b>	analog watchdog 2 high threshold, 0..0xFFFFFF for ADC0/ADC1, 0..0xFF for ADC2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure ADC2 analog watchdog 2 threshold */
```

```
adc_watchdog2_threshold_config(ADC2, 0x40, 0xA0);
```

### adc\_oversample\_mode\_config

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

**Table 3-42. Function adc\_oversample\_mode\_config**

<b>Function name</b>	adc_oversample_mode_config
<b>Function prototype</b>	void adc_oversample_mode_config(uint32_t adc_periph, uint32_t mode, uint16_t shift, uint16_t ratio);
<b>Function descriptions</b>	configure ADC oversample mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
ADCx(x=0,1,2)	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>mode</b>	ADC oversampling mode
ADC_OVERSAMPLING_ALL_CONVERT	all oversampled conversions for a channel are done consecutively after a trigger
ADC_OVERSAMPLING_ONE_CONVERT	each oversampled conversion for a channel needs a trigger
<b>Input parameter{in}</b>	
<b>shift</b>	ADC oversampling shift
ADC_OVERSAMPLING_SHIFT_NONE	no oversampling shift
ADC_OVERSAMPLING_SHIFT_1B	1-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_2B	2-bit oversampling shift
ADC_OVERSAMPLING_SHIFT_3B	3-bit oversampling shift

<code>_SHIFT_3B</code>	
<code>ADC_OVERSAMPLING_SHIFT_4B</code>	4-bit oversampling shift
<code>ADC_OVERSAMPLING_SHIFT_5B</code>	5-bit oversampling shift
<code>ADC_OVERSAMPLING_SHIFT_6B</code>	6-bit oversampling shift
<code>ADC_OVERSAMPLING_SHIFT_7B</code>	7-bit oversampling shift
<code>ADC_OVERSAMPLING_SHIFT_8B</code>	8-bit oversampling shift
<code>ADC_OVERSAMPLING_SHIFT_9B</code>	9-bit oversampling shift, available for ADC0/ADC1
<code>ADC_OVERSAMPLING_SHIFT_10B</code>	10-bit oversampling shift, available for ADC0/ADC1
<code>ADC_OVERSAMPLING_SHIFT_11B</code>	11-bit oversampling shift, available for ADC0/ADC1
<b>Input parameter{in}</b>	
<b>ratio</b>	ADC oversampling ratio, 0..1023 corresponding 1x~1024x for ADC0/ADC1, 0..255 corresponding 1x~256x for ADC2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### adc\_oversample\_mode\_enable

The description of `adc_oversample_mode_enable` is shown as below:

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

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

-	-
Return value	
-	-

Example:

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

```
adc_oversample_mode_enable (ADC0);
```

### adc\_oversample\_mode\_disable

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

**Table 3-44. 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
<i>ADCx(x=0, 1, 2)</i>	ADC peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
adc_oversample_mode_disable (ADC0);
```

### adc\_flag\_get

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

**Table 3-45. Function adc\_flag\_get**

<b>Function name</b>	adc_flag_get
<b>Function prototype</b>	FlagStatus adc_flag_get(uint32_t adc_periph, uint32_t flag);
<b>Function descriptions</b>	get the ADC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0, 1, 2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	

flag	the adc flag bits
<i>ADC_FLAG_WDE0</i>	analog watchdog 0 event flag
<i>ADC_FLAG_EOC</i>	end of sequence conversion flag
<i>ADC_FLAG_EOIC</i>	end of inserted sequence conversion flag
<i>ADC_FLAG_STIC</i>	start flag of inserted sequence
<i>ADC_FLAG_STRC</i>	start flag of routine sequence
<i>ADC_FLAG_ROVF</i>	routine data register overflow flag
<i>ADC_FLAG_WDE1</i>	analog watchdog 1 event flag
<i>ADC_FLAG_WDE2</i>	analog watchdog 2 event flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

```
FlagStatus flag_value;
```

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

### adc\_flag\_clear

The description of `adc_flag_clear` is shown as below:

**Table 3-46. Function `adc_flag_clear`**

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



-	-
Return value	
-	-

Example:

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

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

### adc\_interrupt\_enable

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

**Table 3-47. Function adc\_interrupt\_enable**

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

Example:

```
/* enable ADC0 analog watchdog 0 interrupt */
```

```
adc_interrupt_enable(ADC0, ADC_INT_WDE0);
```

### adc\_interrupt\_disable

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

**Table 3-48. Function adc\_interrupt\_disable**

<b>Function name</b>	adc_interrupt_disable
----------------------	-----------------------

<b>Function prototype</b>	void adc_interrupt_disable(uint32_t adc_periph , uint32_t 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=0,1,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>adc_interrupt</b>	the adc interrupt
<i>ADC_INT_WDE0</i>	analog watchdog 0 interrupt
<i>ADC_INT_EOC</i>	end of sequence conversion interrupt
<i>ADC_INT_EOIC</i>	end of inserted sequence conversion interrupt
<i>ADC_INT_ROVF</i>	routine data register overflow interrupt
<i>ADC_INT_WDE1</i>	analog watchdog 1 interrupt
<i>ADC_INT_WDE2</i>	analog watchdog 2 interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### adc\_interrupt\_flag\_get

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

**Table 3-49. 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, uint32_t int_flag);
<b>Function descriptions</b>	get the ADC interrupt bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adc_periph</b>	ADC peripheral
<i>ADCx(x=0,1,2)</i>	ADC peripheral selection
<b>Input parameter{in}</b>	
<b>int_flag</b>	the adc interrupt bits
<i>ADC_INT_FLAG_WDE0</i> <i>0</i>	analog watchdog 0 interrupt
<i>ADC_INT_FLAG_EOC</i>	end of sequence conversion interrupt
<i>ADC_INT_FLAG_EOIC</i>	end of inserted sequence conversion interrupt
<i>ADC_INT_FLAG_ROV</i>	routine data register overflow interrupt

<i>F</i>	
ADC_INT_FLAG_WDE 1	analog watchdog 1 interrupt
ADC_INT_FLAG_WDE 2	analog watchdog 2 interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

```
FlagStatus flag_value;
```

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

### adc\_interrupt\_flag\_clear

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

**Table 3-50. Function adc\_interrupt\_flag\_clear**

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

-	-
---	---

Example:

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

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

### adc\_sync\_mode\_config

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

**Table 3-51. Function adc\_sync\_mode\_config**

Function name	adc_sync_mode_config
Function prototype	void adc_sync_mode_config(uint32_t sync_mode);
Function descriptions	configure the ADC sync mode
Precondition	-
The called functions	-
Input parameter{in}	
sync_mode	ADC sync mode
ADC_SYNC_MODE_INDEPENDENT	all the ADCs work independently
ADC_DAUL_ROUTINE_PARALLEL_INSERTED_PARALLEL	ADC0 and ADC1 work in combined routine parallel & inserted parallel mode
ADC_DAUL_ROUTINE_PARALLEL_INSERTED_ROTATION	ADC0 and ADC1 work in combined routine parallel & trigger rotation mode
ADC_DAUL_INSERTED_PARALLEL	ADC0 and ADC1 work in inserted parallel mode
ADC_DAUL_ROUTINE_PARALLEL	ADC0 and ADC1 work in routine parallel mode
ADC_DAUL_ROUTINE_FOLLOW_UP	ADC0 and ADC1 work in follow-up mode
ADC_DAUL_INSERTED_TRIGGER_ROTATION	ADC0 and ADC1 work in trigger rotation mode
Output parameter{out}	

-	-
Return value	
-	-

Example:

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

```
adc_sync_mode_config (ADC_DUAL_ROUTINE_PARALLEL_INSERTED_PARALLEL);
```

### adc\_sync\_delay\_config

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

**Table 3-52. Function adc\_interrupt\_disable**

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

Example:

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

```
adc_sync_delay_config (ADC_SYNC_DELAY_5CYCLE);
```

### adc\_sync\_dma\_config

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

**Table 3-53. Function adc\_sync\_dma\_config**

<b>Function name</b>	adc_sync_dma_config
----------------------	---------------------

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

Example:

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

```
adc_sync_dma_config (ADC_SYNC_DMA_MODE0);
```

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

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

**Table 3-54. Function adc\_sync\_dma\_request\_after\_last\_enable**

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

Example:

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

```
adc_sync_dma_request_after_last_enable();
```

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

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

**Table 3-55. Function adc\_sync\_dma\_request\_after\_last\_disable**

<b>Function name</b>	adc_sync_dma_request_after_last_disable
<b>Function prototype</b>	void adc_sync_dma_request_after_last_disable (void);
<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>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_sync_dma_request_after_last_disable();
```

## adc\_sync\_master\_adc\_routine\_data0\_read

The description of adc\_sync\_master\_adc\_routine\_data0\_read is shown as below:

**Table 3-56. Function adc\_sync\_master\_adc\_routine\_data0\_read**

<b>Function name</b>	adc_sync_master_adc_routine_data0_read
<b>Function prototype</b>	uint16_t adc_sync_master_adc_routine_data0_read (void);
<b>Function descriptions</b>	read ADC sync master adc routine data register 0
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint16_t	sync routine data 0

Example:

```
/* read ADC sync master adc routine data register0 */
```

```
adc_sync_master_adc_routine_data0_read ();
```

## adc\_sync\_slave\_adc\_routine\_data0\_read

The description of adc\_sync\_slave\_adc\_routine\_data0\_read is shown as below:

**Table 3-57. Function adc\_sync\_slave\_adc\_routine\_data0\_read**

<b>Function name</b>	adc_sync_slave_adc_routine_data0_read
<b>Function prototype</b>	uint16_t adc_sync_slave_adc_routine_data0_read (void);
<b>Function descriptions</b>	read ADC sync slave adc routine data register 0
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint16_t	sync routine data 0

Example:

```
/* read ADC sync slave adc routine data register0 */
adc_sync_slave_adc_routine_data0_read ();
```

## adc\_sync\_routine\_data1\_read

The description of adc\_sync\_routine\_data1\_read is shown as below:

**Table 3-58. Function adc\_sync\_routine\_data1\_read**

<b>Function name</b>	adc_sync_routine_data1_read
<b>Function prototype</b>	uint32_t adc_sync_routine_data1_read(void);
<b>Function descriptions</b>	read ADC sync routine data register 1
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	sync routine data 1

Example:

```
/* read ADC sync routine data register 1 */
adc_sync_routine_data1_read ();
```



### 3.3. CAN

CAN bus (Controller Area Network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer. The CAN interface supports the CAN 2.0A/B protocol, ISO 11898-1:2015 and BOSCH CAN FD specification. The CAN registers are listed in chapter [3.3.1](#), the CAN firmware functions are introduced in chapter [3.3.2](#).

#### 3.3.1. Descriptions of Peripheral registers

CAN registers are listed in the table shown as below:

**Table 3-59. CAN Registers**

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

Registers	Descriptions
CAN_PN_RWMxI (x = 0..3)	Pretended Networking mode received wakeup mailbox x identifier register
CAN_PN_RWMxD0 (x = 0..3)	Pretended Networking mode received wakeup mailbox x data 0 register
CAN_PN_RWMxD1 (x = 0..3)	Pretended Networking mode received wakeup mailbox x data 1 register
CAN_FDCTL	CAN FD control register
CAN_FDBT	CAN bit timing register
CAN_CRCCFD	CAN CRC for classical and FD frame register

### 3.3.2. Descriptions of Peripheral functions

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

**Table 3-60. CAN firmware function**

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

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

### Structure can\_error\_counter\_struct

**Table 3-61. Structure can\_error\_counter\_struct**

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

Member name	Function description
tx_errcnt	transmit error count defined by the CAN standard

### Structure can\_parameter\_struct

**Table 3-62. Structure can\_parameter\_struct**

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

### Structure can\_mailbox\_descriptor\_struct

**Table 3-63. Structure can\_mailbox\_descriptor\_struct**

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

Member name	Function description
data[64]	data
data_bytes	data bytes
padding	FD mode padding data

### Structure can\_rx\_fifo\_struct

**Table 3-64. Structure can\_rx\_fifo\_struct**

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

### Structure can\_fd\_parameter\_struct

**Table 3-65. Structure can\_fd\_parameter\_struct**

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

### Structure can\_rx\_fifo\_id\_filter\_struct

**Table 3-66. Structure can\_rx\_fifo\_id\_filter\_struct**

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

### Structure can\_fifo\_parameter\_struct

**Table 3-67. Structure can\_fifo\_parameter\_struct**

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

dma_enable	DMA enable
filter_format_and_number	FIFO ID filter format and number
fifo_public_filter	FIFO ID public filter

### Structure can\_pn\_mode\_filter\_struct

**Table 3-68. Structure can\_pn\_mode\_filter\_struct**

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

### Structure can\_pn\_mode\_config\_struct

**Table 3-69. Structure can\_pn\_mode\_config\_struct**

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

### Structure can\_crc\_struct

**Table 3-70. Structure can\_crc\_struct**

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

## Enum can\_interrupt\_enum

**Table 3-71. Enum can\_interrupt\_enum**

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

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

### Enum can\_flag\_enum

Table 3-72. Enum can\_flag\_enum

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



Member name	Function description
RR	
CAN_FLAG_BIT_DOMINANT_ERR	bit dominant error flag
CAN_FLAG_BIT_RECESSIVE_ERR	bit recessive error flag
CAN_FLAG_SYNC_ERR	synchronization flag
CAN_FLAG_BUSOFF_RECOVERY	bus off recovery flag
CAN_FLAG_ERR_SUMMARY_FD	FD error summary flag
CAN_FLAG_ERR_OVERRUN	error overrun flag
CAN_FLAG_STUFF_ERR_FD	stuff error in FD data phase flag
CAN_FLAG_FORM_ERR_FD	form error in FD data phase flag
CAN_FLAG_CRC_ERR_FD	CRC error in FD data phase flag
CAN_FLAG_BIT_DOMINANT_ERR_FD	bit dominant error in FD data phase flag
CAN_FLAG_BIT_RECESSIVE_ERR_FD	bit recessive error in FD data phase flag
CAN_FLAG_MB0	mailbox 0 flag
CAN_FLAG_MB1	mailbox 1 flag
CAN_FLAG_MB2	mailbox 2 flag
CAN_FLAG_MB3	mailbox 3 flag
CAN_FLAG_MB4	mailbox 4 flag
CAN_FLAG_MB5	mailbox 5 flag
CAN_FLAG_MB6	mailbox 6 flag
CAN_FLAG_MB7	mailbox 7 flag
CAN_FLAG_MB8	mailbox 8 flag
CAN_FLAG_MB9	mailbox 9 flag
CAN_FLAG_MB10	mailbox 10 flag
CAN_FLAG_MB11	mailbox 11 flag
CAN_FLAG_MB12	mailbox 12 flag
CAN_FLAG_MB13	mailbox 13 flag
CAN_FLAG_MB14	mailbox 14 flag
CAN_FLAG_MB15	mailbox 15 flag
CAN_FLAG_MB16	mailbox 16 flag
CAN_FLAG_MB17	mailbox 17 flag

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

### Enum can\_interrupt\_flag\_enum

**Table 3-73. Enum can\_interrupt\_flag\_enum**

Member name	Function description
CAN_INT_FLAG_ERROR_SUMMARY	error summary interrupt flag
CAN_INT_FLAG_BUSOFF	bus off interrupt flag
CAN_INT_FLAG_RX_WARNING	receive warning interrupt flag
CAN_INT_FLAG_TX_WARNING	transmit warning interrupt flag
CAN_INT_FLAG_BUSOFF_RECOVERY	bus off recovery interrupt flag
CAN_INT_FLAG_FD_ERROR_SUMMARY	fd error summary interrupt flag

Member name	Function description
RR_SUMMARY_FD	
CAN_INT_FLAG_M B0	mailbox 0 interrupt flag
CAN_INT_FLAG_M B1	mailbox 1 interrupt flag
CAN_INT_FLAG_M B2	mailbox 2 interrupt flag
CAN_INT_FLAG_M B3	mailbox 3 interrupt flag
CAN_INT_FLAG_M B4	mailbox 4 interrupt flag
CAN_INT_FLAG_M B5	mailbox 5 interrupt flag
CAN_INT_FLAG_M B6	mailbox 6 interrupt flag
CAN_INT_FLAG_M B7	mailbox 7 interrupt flag
CAN_INT_FLAG_M B8	mailbox 8 interrupt flag
CAN_INT_FLAG_M B9	mailbox 9 interrupt flag
CAN_INT_FLAG_M B10	mailbox 10 interrupt flag
CAN_INT_FLAG_M B11	mailbox 11 interrupt flag
CAN_INT_FLAG_M B12	mailbox 12 interrupt flag
CAN_INT_FLAG_M B13	mailbox 13 interrupt flag
CAN_INT_FLAG_M B14	mailbox 14 interrupt flag
CAN_INT_FLAG_M B15	mailbox 15 interrupt flag
CAN_INT_FLAG_M B16	mailbox 16 interrupt flag
CAN_INT_FLAG_M B17	mailbox 17 interrupt flag
CAN_INT_FLAG_M B18	mailbox 18 interrupt flag
CAN_INT_FLAG_M B19	mailbox 19 interrupt flag
CAN_INT_FLAG_M	mailbox 20 interrupt flag

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

### Enum can\_operation\_modes\_enum

Table 3-74. Enum can\_operation\_modes\_enum

Member name	Function description
CAN_NORMAL_MO DE	normal mode
CAN_MONITOR_M ODE	monitor mode
CAN_LOOPBACK_	loopback mode

Member name	Function description
SILENT_MODE	
CAN_INACTIVE_MODE	inactive mode
CAN_DISABLE_MODE	disable mode
CAN_PN_MODE	Pretended Networking mode

### Enum can\_struct\_type\_enum

**Table 3-75. Enum can\_struct\_type\_enum**

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

### Enum can\_error\_state\_enum

**Table 3-76. Enum can\_error\_state\_enum**

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

### can\_deinit

The description of can\_deinit is shown as below:

**Table 3-77. Function can\_deinit**

Function name	can_deinit
Function prototype	void can_deinit(uint32_t can_periph);
Function descriptions	deinitialize CAN
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
can_periph	CAN peripheral

CANx(x=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### can\_software\_reset

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

**Table 3-78. Function can\_software\_reset**

Function name	can_software_reset
Function prototype	ErrStatus can_software_reset(uint32_t can_periph);
Function descriptions	reset CAN internal state machines and CAN registers
Precondition	-
The called functions	
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;

/* reset CAN0 */

err = can_software_reset(CAN0);
```

### can\_init

The description of can\_init is shown as below:

**Table 3-79. Function can\_init**

Function name	can_init
Function prototype	ErrStatus can_init(uint32_t can_periph, can_parameter_struct* can_parameter_init);
Function descriptions	CAN module initialization
Precondition	-
The called functions	-

Input parameter{in}	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
Input parameter{in}	
<b>can_parameter_init</b>	Refers to <a href="#">Table 3-62. Structure can_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
can_parameter_struct can_parameter;

ErrStatus err;

.....

/* initialize CAN */

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

### can\_struct\_para\_init

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

**Table 3-80. Function can\_struct\_para\_init**

<b>Function name</b>	can_struct_para_init
<b>Function prototype</b>	void can_struct_para_init(can_struct_type_enum type, void* p_struct);
<b>Function descriptions</b>	initialize CAN parameter structure with a default value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>type</b>	Refers to enum <a href="#">Table 3-75. Enum can_struct_type_enum</a>
Input parameter{in}	
<b>p_struct</b>	the pointer of the specific struct
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_parameter_struct can_parameter;

/* initialize CAN */

can_struct_para_init(CAN_INIT_STRUCT, &can_parameter);
```

## can\_private\_filter\_config

The description of can\_private\_filter\_config is shown as below:

**Table 3-81. Function can\_private\_filter\_config**

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

Example:

```
/* configure CAN0 mailbox 0 private filter */
can_private_filter_config(CAN0, 0, 0xFFFFFFFF);
```

## can\_operation\_mode\_enter

The description of can\_operation\_mode\_enter is shown as below:

**Table 3-82. Function can\_operation\_mode\_enter**

<b>Function name</b>	can_operation_mode_enter
<b>Function prototype</b>	ErrStatus can_operation_mode_enter(uint32_t can_periph, can_operation_modes_enum mode);
<b>Function descriptions</b>	enter the corresponding mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	



<b>mode</b>	Refers to enum <a href="#">Table 3-74. Enum can_operation_modes_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

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

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

### can\_operation\_mode\_get

The description of can\_operation\_mode\_get is shown as below:

**Table 3-83. Function can\_operation\_mode\_get**

<b>Function name</b>	can_operation_mode_get
<b>Function prototype</b>	can_operation_modes_enum can_operation_mode_get(uint32_t can_periph);
<b>Function descriptions</b>	get operation mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>can_operation_modes_enum</b>	Refers to enum <a href="#">Table 3-74. Enum can_operation_modes_enum</a>

Example:

```
can_operation_modes_enum mode;
```

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

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

### can\_inactive\_mode\_exit

The description of can\_inactive\_mode\_exit is shown as below:

**Table 3-84. Function can\_inactive\_mode\_exit**

<b>Function name</b>	can_inactive_mode_exit
<b>Function prototype</b>	ErrStatus can_inactive_mode_exit(uint32_t can_periph);

<b>Function descriptions</b>	exit inactive mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

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

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

### can\_pn\_mode\_exit

The description of can\_pn\_mode\_exit is shown as below:

**Table 3-85. Function can\_pn\_mode\_exit**

<b>Function name</b>	can_pn_mode_exit
<b>Function prototype</b>	ErrStatus can_pn_mode_exit(uint32_t can_periph);
<b>Function descriptions</b>	exit Pretended Networking mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

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

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

### can\_fd\_config

The description of can\_fd\_config is shown as below:

Table 3-86. Function can\_fd\_config

Function name	can_fd_config
Function prototype	void can_fd_config(uint32_t can_periph, can_fd_parameter_struct *can_fd_para_init);
Function descriptions	can FD initialize
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
Input parameter{in}	
can_fd_para_init	Refers to structure <a href="#">Table 3-65. Structure can_fd_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_fd_parameter_struct fd_parameter;

/* FD parameter configurations */

.....

can_fd_config(CAN0, &fd_parameter);
```

### can\_bitrate\_switch\_enable

The description of can\_bitrate\_switch\_enable is shown as below:

Table 3-87. Function can\_bitrate\_switch\_enable

Function name	can_bitrate_switch_enable
Function prototype	void can_bitrate_switch_enable(uint32_t can_periph);
Function descriptions	enable bit rate switching
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CAN0 bit rate switching */
```

```
can_bitrate_switch_enable(CAN0);
```

### can\_bitrate\_switch\_disable

The description of can\_bitrate\_switch\_disable is shown as below:

**Table 3-88. Function can\_bitrate\_switch\_disable**

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

Example:

```
/* disable CAN0 bit rate switching */
```

```
can_bitrate_switch_disable(CAN0);
```

### can\_tdc\_get

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

**Table 3-89. Function can\_tdc\_get**

<b>Function name</b>	can_tdc_get
<b>Function prototype</b>	uint32_t can_tdc_get(uint32_t can_periph);
<b>Function descriptions</b>	get transmitter delay compensation value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	0 - 0x3F

Example:

```
uint32_t tdc;
```

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

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

### can\_tdc\_enable

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

**Table 3-90. Function can\_tdc\_enable**

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

Example:

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

```
can_tdc_enable(CAN0);
```

### can\_tdc\_disable

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

**Table 3-91. Function can\_tdc\_disable**

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

Example:

```
/* disable transmitter delay compensation */
```

```
can_tdc_disable(CAN0);
```

### can\_rx\_fifo\_config

The description of can\_rx\_fifo\_config is shown as below:

**Table 3-92. Function can\_rx\_fifo\_config**

<b>Function name</b>	can_rx_fifo_config
<b>Function prototype</b>	void can_rx_fifo_config(uint32_t can_periph, can_fifo_parameter_struct *can_fifo_para_init);
<b>Function descriptions</b>	configure rx FIFO
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>can_fifo_para_init</b>	Refers to structure <a href="#">Table 3-67. Structure can_fifo_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
can_fifo_parameter_struct fifo_struct;
```

```
/* configure rx FIFO */
```

```
.....
```

```
can_rx_fifo_config(CAN0, &fifo_struct);
```

### can\_rx\_fifo\_filter\_table\_config

The description of can\_rx\_fifo\_filter\_table\_config is shown as below:

**Table 3-93. Function can\_rx\_fifo\_filter\_table\_config**

<b>Function name</b>	can_rx_fifo_filter_table_config
<b>Function prototype</b>	void can_rx_fifo_filter_table_config(uint32_t can_periph, can_rx_fifo_id_filter_struct id_filter_table[]);
<b>Function descriptions</b>	configure rx FIFO filter table
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral

CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	
id_filter_table	Refers to structure <a href="#">Table 3-66. Structure can_rx_fifo_id_filter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
can_rx_fifo_id_filter_struct id_filter_table[104];

/* configure rx FIFO filter table */

.....

can_rx_fifo_filter_table_config(CAN0, id_filter_table);
```

### can\_rx\_fifo\_read

The description of can\_rx\_fifo\_read is shown as below:

**Table 3-94. Function can\_rx\_fifo\_read**

<b>Function name</b>	can_rx_fifo_read
<b>Function prototype</b>	void can_rx_fifo_read(uint32_t can_periph, can_rx_fifo_struct *rx_fifo);
<b>Function descriptions</b>	read rx FIFO data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Output parameter{out}</b>	
rx_fifo	Refers to structure <a href="#">Table 3-64. Structure can_rx_fifo_struct</a>
<b>Return value</b>	
-	-

Example:

```
can_rx_fifo_struct rx_fifo;

/* read rx FIFO data */

can_rx_fifo_read(CAN0, &rx_fifo);
```

### can\_rx\_fifo\_filter\_matching\_number\_get

The description of can\_rx\_fifo\_filter\_matching\_number\_get is shown as below:

**Table 3-95. Function can\_rx\_fifo\_filter\_matching\_number\_get**

<b>Function name</b>	can_rx_fifo_filter_matching_number_get
----------------------	--

<b>Function prototype</b>	uint32_t can_rx_fifo_filter_matching_number_get(uint32_t can_periph);
<b>Function descriptions</b>	get rx FIFO filter matching number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0,1,2)</i>	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	0-416

Example:

```
uint32_t number;
```

```
/* get rx FIFO filter matching number */
```

```
number = can_rx_fifo_filter_matching_number_get(CAN0);
```

### can\_rx\_fifo\_clear

The description of can\_rx\_fifo\_clear is shown as below:

**Table 3-96. Function can\_rx\_fifo\_clear**

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

Example:

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

```
can_rx_fifo_clear(CAN0);
```

### can\_ram\_address\_get

The description of can\_ram\_address\_get is shown as below:



Table 3-97. Function can\_ram\_address\_get

Function name	can_ram_address_get
Function prototype	uint32_t* can_ram_address_get(uint32_t can_periph, uint32_t index);
Function descriptions	get mailbox RAM address
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Output parameter{out}	
-	-
Return value	
uint32_t	0-0xFFFFFFFF

Example:

```
uint32_t address;
```

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

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

### can\_mailbox\_config

The description of can\_mailbox\_config is shown as below:

Table 3-98. Function can\_mailbox\_config

Function name	can_mailbox_config
Function prototype	void can_mailbox_config(uint32_t can_periph, uint32_t index, can_mailbox_descriptor_struct *mdpara);
Function descriptions	config mailbox
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure <a href="#">Table 3-63. Structure can_mailbox_descriptor_struct</a>
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
can_mailbox_descriptor_struct transmit_message;
```

```
.....
```

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

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

### can\_mailbox\_transmit\_abort

The description of can\_mailbox\_transmit\_abort is shown as below:

**Table 3-99. Function can\_mailbox\_transmit\_abort**

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

Example:

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

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

### can\_mailbox\_transmit\_inactive

The description of can\_mailbox\_transmit\_inactive is shown as below:

**Table 3-100. Function can\_mailbox\_transmit\_inactive**

Function name	can_mailbox_transmit_inactive
Function prototype	void can_mailbox_transmit_inactive(uint32_t can_periph, uint32_t index);
Function descriptions	inactive transmit mailbox
Precondition	-

The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

### can\_mailbox\_receive\_data\_read

The description of can\_mailbox\_receive\_data\_read is shown as below:

**Table 3-101. Function can\_mailbox\_receive\_data\_read**

Function name	can_mailbox_receive_data_read
Function prototype	ErrStatus can_mailbox_receive_data_read(uint32_t can_periph, uint32_t index, can_mailbox_descriptor_struct *mdpara);
Function descriptions	read receive mailbox data
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
Input parameter{in}	
index	mailbox index
0-31	mailbox index selection
Input parameter{in}	
mdpara	Refers to structure <a href="#">Table 3-63. Structure can_mailbox_descriptor_struct</a>
Output parameter{out}	
-	-
Return value	
ErrStatus	ERROR or SUCCESS

Example:

```
ErrStatus err;
```

```
can_mailbox_descriptor_struct receive_message;
```

.....

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

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

### can\_mailbox\_receive\_lock

The description of can\_mailbox\_receive\_lock is shown as below:

**Table 3-102. Function can\_mailbox\_receive\_lock**

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

Example:

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

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

### can\_mailbox\_receive\_unlock

The description of can\_mailbox\_receive\_unlock is shown as below:

**Table 3-103. Function can\_mailbox\_receive\_unlock**

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

Return value	
-	-

Example:

```
/* unlock the receive mailbox */
```

```
can_mailbox_receive_unlock(CAN0);
```

### can\_mailbox\_receive\_inactive

The description of can\_mailbox\_receive\_inactive is shown as below:

**Table 3-104. Function can\_mailbox\_receive\_inactive**

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

Example:

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

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

### can\_mailbox\_code\_get

The description of can\_mailbox\_code\_get is shown as below:

**Table 3-105. Function can\_mailbox\_code\_get**

<b>Function name</b>	can_mailbox_code_get
<b>Function prototype</b>	uint32_t can_mailbox_code_get(uint32_t can_periph, uint32_t index);
<b>Function descriptions</b>	get mailbox code value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral

<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>index</b>	mailbox index
<i>0-31</i>	mailbox index selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	0-0xF

Example:

```
uint32_t code;
```

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

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

### can\_error\_counter\_config

The description of can\_error\_counter\_config is shown as below:

**Table 3-106. Function can\_error\_counter\_config**

<b>Function name</b>	can_error_counter_config
<b>Function prototype</b>	void can_error_counter_config(uint32_t can_periph, can_error_counter_struct *errcnt_struct);
<b>Function descriptions</b>	configure error counter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>errcnt_struct</b>	Refers to structure <a href="#">Table 3-61. Structure can error counter struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
can_error_counter_struct err_struct;
```

```
.....
```

```
/* configure error counter */
```

```
can_error_counter_config(CAN0, &err_struct);
```

## can\_error\_counter\_get

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

**Table 3-107. Function can\_error\_counter\_get**

<b>Function name</b>	can_error_counter_get
<b>Function prototype</b>	void can_error_counter_get(uint32_t can_periph, can_error_counter_struct *errcnt_struct);
<b>Function descriptions</b>	get error count
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>errcnt_struct</b>	Refers to structure <a href="#">Table 3-61. Structure can_error_counter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
can_error_counter_struct err_struct;

/* get error count */

can_error_counter_get(CAN0, &err_struct);
```

## can\_error\_state\_get

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

**Table 3-108. Function can\_error\_state\_get**

<b>Function name</b>	can_error_state_get
<b>Function prototype</b>	can_error_state_enum can_error_state_get(uint32_t can_periph);
<b>Function descriptions</b>	get error state indicator
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>can_error_state_enum</b>	Refers to enum <a href="#">Table 3-76. Enum can_error_state_enum</a>

Example:

```
can_error_state_enum error_state;

/* get error state indicator */

error_state = can_error_state_get(CAN0);
```

### can\_crc\_get

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

**Table 3-109. Function can\_crc\_get**

<b>Function name</b>	can_crc_get
<b>Function prototype</b>	void can_crc_get(uint32_t can_periph, can_crc_struct *crc_struct);
<b>Function descriptions</b>	get mailbox CRC value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<b>CANx(x=0,1,2)</b>	CAN peripheral selection
<b>Return value</b>	
<b>crc_struct</b>	Refers to structure <a href="#">Table 3-70. Structure can_crc_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
can_crc_struct crc_struct;

/* get mailbox CRC value */

can_crc_get(CAN0, &crc_struct);
```

### can\_pn\_mode\_config

The description of can\_pn\_mode\_config is shown as below:

**Table 3-110. Function can\_pn\_mode\_config**

<b>Function name</b>	can_pn_mode_config
<b>Function prototype</b>	void can_pn_mode_config(uint32_t can_periph, can_pn_mode_config_struct *pnmod_config);
<b>Function descriptions</b>	configure Pretended Networking mode parameter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral



CANx(x=0, 1, 2)	CAN peripheral selection
Return value	
pnmod_config	Refers to structure <a href="#">Table 3-69. Structure can_pn_mode_config_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_pn_mode_config_struct pn_struct;
```

```
.....
```

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

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

### can\_pn\_mode\_filter\_config

The description of can\_pn\_mode\_filter\_config is shown as below:

**Table 3-111. Function can\_pn\_mode\_filter\_config**

Function name	can_pn_mode_filter_config
Function prototype	void can_pn_mode_filter_config(uint32_t can_periph, can_pn_mode_filter_struct *expect, can_pn_mode_filter_struct *filter);
Function descriptions	configure pn mode filter
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
Return value	
expect	Refers to structure <a href="#">Table 3-68. Structure can_pn_mode_filter_struct</a>
Return value	
filter	Refers to structure <a href="#">Table 3-68. Structure can_pn_mode_filter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_pn_mode_filter_struct pn_filter[2];
```

```
.....
```

```
/* configure pn mode filter */
```

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

### can\_pn\_mode\_num\_of\_match\_get

The description of can\_pn\_mode\_num\_of\_match\_get is shown as below:

**Table 3-112. Function can\_pn\_mode\_num\_of\_match\_get**

<b>Function name</b>	can_pn_mode_num_of_match_get
<b>Function prototype</b>	int32_t can_pn_mode_num_of_match_get(uint32_t can_periph);
<b>Function descriptions</b>	get matching message counter of Pretended Networking mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
int32_t	0-255 or -1

Example:

```
int32_t counter;
```

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

```
counter = can_pn_mode_num_of_match_get(CAN0);
```

### can\_pn\_mode\_data\_read

The description of can\_pn\_mode\_data\_read is shown as below:

**Table 3-113. Function can\_pn\_mode\_data\_read**

<b>Function name</b>	can_pn_mode_data_read
<b>Function prototype</b>	void can_pn_mode_data_read(uint32_t can_periph, uint32_t index, can_mailbox_descriptor_struct *mdpara);
<b>Function descriptions</b>	get matching message
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>index</b>	mailbox index
0-31	mailbox index selection
<b>Input parameter{in}</b>	
<b>mdpara</b>	Refers to structure <a href="#">Table 3-63. Structure can_mailbox_descriptor_struct</a>

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
can_mailbox_descriptor_struct mb_para;

/* get matching message */

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

### can\_self\_reception\_enable

The description of can\_self\_reception\_enable is shown as below:

**Table 3-114. Function can\_self\_reception\_enable**

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

Example:

```
/* enable self reception */

can_self_reception_enable(CAN0);
```

### can\_self\_reception\_disable

The description of can\_self\_reception\_disable is shown as below:

**Table 3-115. Function can\_self\_reception\_disable**

Function name	can_self_reception_disable
Function prototype	void can_self_reception_disable(uint32_t can_periph);
Function descriptions	disable self reception
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral

CANx(x=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable self reception */
```

```
can_self_reception_disable(CAN0);
```

### can\_transmit\_abort\_enable

The description of can\_transmit\_abort\_enable is shown as below:

**Table 3-116. Function can\_transmit\_abort\_enable**

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

Example:

```
/* enable transmit abort */
```

```
can_transmit_abort_enable(CAN0);
```

### can\_transmit\_abort\_disable

The description of can\_transmit\_abort\_disable is shown as below:

**Table 3-117. Function can\_transmit\_abort\_disable**

Function name	can_transmit_abort_disable
Function prototype	void can_transmit_abort_disable(uint32_t can_periph);
Function descriptions	disable transmit abort
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral

CANx(x=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable transmit abort */
```

```
can_transmit_abort_disable(CAN0);
```

### can\_auto\_busoff\_recovery\_enable

The description of can\_auto\_busoff\_recovery\_enable is shown as below:

**Table 3-118. Function can\_auto\_busoff\_recovery\_enable**

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

Example:

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

```
can_auto_busoff_recovery_enable(CAN0);
```

### can\_auto\_busoff\_recovery\_disable

The description of can\_auto\_busoff\_recovery\_disable is shown as below:

**Table 3-119. Function can\_auto\_busoff\_recovery\_disable**

Function name	can_auto_busoff_recovery_disable
Function prototype	void can_auto_busoff_recovery_disable(uint32_t can_periph);
Function descriptions	disable auto bus off recovery mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral

CANx(x=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable auto bus off recovery mode */
can_auto_busoff_recovery_disable(CAN0);
```

### can\_time\_sync\_enable

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

**Table 3-120. Function can\_time\_sync\_enable**

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

Example:

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

### can\_time\_sync\_disable

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

**Table 3-121. Function can\_time\_sync\_disable**

Function name	can_time_sync_disable
Function prototype	void can_time_sync_disable(uint32_t can_periph);
Function descriptions	disable time sync mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral

CANx(x=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### can\_edge\_filter\_mode\_enable

The description of can\_edge\_filter\_mode\_enable is shown as below:

**Table 3-122. Function can\_edge\_filter\_mode\_enable**

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

Example:

```
/* enable edge filter mode */
can_edge_filter_mode_enable(CAN0);
```

### can\_edge\_filter\_mode\_disable

The description of can\_edge\_filter\_mode\_disable is shown as below:

**Table 3-123. Function can\_edge\_filter\_mode\_disable**

Function name	can_edge_filter_mode_disable
Function prototype	void can_edge_filter_mode_disable(uint32_t can_periph);
Function descriptions	disable edge filter mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral

CANx(x=0,1,2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable edge filter mode */
can_edge_filter_mode_disable(CAN0);
```

### can\_ped\_mode\_enable

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

**Table 3-124. Function can\_ped\_mode\_enable**

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

Example:

```
/* enable protocol exception detection mode */
can_ped_mode_enable(CAN0);
```

### can\_ped\_mode\_disable

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

**Table 3-125. Function can\_ped\_mode\_disable**

Function name	can_ped_mode_disable
Function prototype	void can_ped_mode_disable(uint32_t can_periph);
Function descriptions	disable protocol exception detection mode
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral



CANx(x=0, 1, 2)	CAN peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
can_ped_mode_disable(CAN0);
```

### can\_arbitration\_delay\_bits\_config

The description of can\_arbitration\_delay\_bits\_config is shown as below:

**Table 3-126. Function can\_arbitration\_delay\_bits\_config**

Function name	can_arbitration_delay_bits_config
Function prototype	void can_arbitration_delay_bits_config(uint32_t can_periph, uint32_t delay_bits);
Function descriptions	configure arbitration delay bits
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
Input parameter{in}	
delay_bits	delay bits
0-31	delay bits selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure arbitration delay bits */
```

```
can_arbitration_delay_bits_config(CAN0, 2);
```

### can\_bsp\_mode\_config

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

**Table 3-127. Function can\_bsp\_mode\_config**

Function name	can_bsp_mode_config
Function prototype	void can_bsp_mode_config(uint32_t can_periph, uint32_t sampling_mode);
Function descriptions	configure bit sampling mode

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>sampling_mode</b>	bsp sample mode
<i>CAN_BSP_MODE_ON E_SAMPLE</i>	one sample for received bit
<i>CAN_BSP_MODE_TR HEE_SAMPLES</i>	three samples for received bit
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure bit sampling mode */
```

```
can_bsp_mode_config(CAN0, CAN_BSP_MODE_ONE_SAMPLE);
```

### can\_flag\_get

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

**Table 3-128. Function can\_flag\_get**

<b>Function name</b>	can_flag_get
<b>Function prototype</b>	FlagStatus can_flag_get(uint32_t can_periph, can_flag_enum flag);
<b>Function descriptions</b>	get CAN flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>flag</b>	Refers to enum <a href="#">Table 3-72. Enum can_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
FlagStatus flag;
```

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

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

## can\_flag\_clear

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

**Table 3-129. Function can\_flag\_clear**

<b>Function name</b>	can_flag_clear
<b>Function prototype</b>	void can_flag_clear(uint32_t can_periph, can_flag_enum flag);
<b>Function descriptions</b>	clear CAN flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>flag</b>	Refers to enum <a href="#">Table 3-72. Enum can_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

## can\_interrupt\_enable

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

**Table 3-130. Function can\_interrupt\_enable**

<b>Function name</b>	can_interrupt_enable
<b>Function prototype</b>	void can_interrupt_enable(uint32_t can_periph, can_interrupt_enum interrupt);
<b>Function descriptions</b>	enable CAN interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
CANx(x=0,1,2)	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>interrupt</b>	Refers to enum <a href="#">Table 3-71. Enum can_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

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

### can\_interrupt\_disable

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

**Table 3-131. Function can\_interrupt\_disable**

<b>Function name</b>	can_interrupt_disable
<b>Function prototype</b>	void can_interrupt_disable(uint32_t can_periph, can_interrupt_enum interrupt);
<b>Function descriptions</b>	disable CAN interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral
<i>CANx(x=0, 1, 2)</i>	CAN peripheral selection
<b>Input parameter{in}</b>	
<b>interrupt</b>	Refers to enum <a href="#">Table 3-71. Enum can_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### can\_interrupt\_flag\_get

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

**Table 3-132. Function can\_interrupt\_flag\_get**

<b>Function name</b>	can_interrupt_flag_get
<b>Function prototype</b>	FlagStatus can_interrupt_flag_get(uint32_t can_periph, can_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	get CAN interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_periph</b>	CAN peripheral

CANx(x=0, 1, 2)	CAN peripheral selection
Input parameter{in}	
int_flag	Refers to enum <a href="#">Table 3-73. Enum can_interrupt_flag_enum</a>
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
FlagStatus int_flag;
```

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

```
int_flag = can_interrupt_flag_get(CAN0, CAN_INT_FLAG_FIFO_AVAILABLE);
```

### can\_interrupt\_flag\_clear

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

**Table 3-133. Function can\_interrupt\_flag\_clear**

Function name	can_interrupt_flag_clear
Function prototype	void can_interrupt_flag_clear(uint32_t can_periph, can_interrupt_flag_enum int_flag);
Function descriptions	clear CAN interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
can_periph	CAN peripheral
CANx(x=0, 1, 2)	CAN peripheral selection
Input parameter{in}	
int_flag	Refers to enum <a href="#">Table 3-73. Enum can_interrupt_flag_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

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

## 3.4. 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 [3.4.1](#), the CMP firmware functions are introduced in chapter [3.4.2](#).

### 3.4.1. Descriptions of Peripheral registers

CMP registers are listed in the table shown as below:

**Table 3-134. CMP registers**

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

### 3.4.2. Descriptions of Peripheral functions

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

**Table 3-135. CMP firmware function**

Function name	Function description
cmp_deinit	CMP deinit
cmp_mode_init	CMP mode init
cmp_noninverting_input_select	CMP noninverting input select
cmp_output_init	CMP output init
cmp_output_mux_config	config comparator output port
cmp_blanking_init	CMP output blanking function init
cmp_enable	enable CMP
cmp_disable	disable CMP
cmp_window_enable	enable the window mode
cmp_window_disable	disable the window mode
cmp_lock_enable	lock the CMP
cmp_voltage_scaler_enable	enable the voltage scaler
cmp_voltage_scaler_disable	disable the voltage scaler
cmp_scaler_bridge_enable	enable the scaler bridge
cmp_scaler_bridge_disable	disable the scaler bridge
cmp_output_level_get	get output level
cmp_flag_get	get CMP flag
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 3-136. Enum cmp\_enum

Member name	Function description
CMP0	comparator 0
CMP1	comparator 1

## cmp\_deinit

The description of cmp\_deinit is shown as below:

Table 3-137. Function cmp\_deinit

Function name	cmp_deinit
Function prototype	void cmp_deinit(cmp_enum cmp_periph);
Function descriptions	CMP deinit
Precondition	-
The called functions	rcu_periph_reset_enable / rcu_periph_reset_disable
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

## cmp\_mode\_init

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

Table 3-138. Function cmp\_mode\_init

Function name	cmp_mode_init
Function prototype	void cmp_mode_init(cmp_enum cmp_periph, uint32_t operating_mode, uint32_t inverting_input, uint32_t output_hysteresis);
Function descriptions	CMP mode init
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
Input parameter{in}	
operating_mode	operating mode
CMP_MODE_HIGHSP EED	high speed mode

<i>CMP_MODE_MIDDLE SPEED</i>	medium speed mode
<i>CMP_MODE_VERYLO WSPEED</i>	very-low speed mode
<b>Input parameter{in}</b>	
<b>inverting_input</b>	inverting input
<i>CMP_INVERTING_INP UT_1_4VREFINT</i>	VREFINT *1/4 input
<i>CMP_INVERTING_INP UT_1_2VREFINT</i>	VREFINT *1/2 input
<i>CMP_INVERTING_INP UT_3_4VREFINT</i>	VREFINT *3/4 input
<i>CMP_INVERTING_INP UT_VREFINT</i>	VREFINT input
<i>CMP_INVERTING_INP UT_DAC0_OUT0</i>	PA4 (DAC) input
<i>CMP_INVERTING_INP UT_DAC0_OUT1</i>	PA5 (DAC) input
<i>CMP_INVERTING_INP UT_PB1_PE10</i>	PB1 for CMP0 or PE10 for CMP1 as inverting input
<i>CMP_INVERTING_INP UT_PC4_PE7</i>	PC4 for CMP0 or PE7 for CMP1 as inverting input
<b>Input parameter{in}</b>	
<b>output_hysteresis</b>	hysteresis level
<i>CMP_HYSTERESIS_N O</i>	output no hysteresis
<i>CMP_HYSTERESIS_L OW</i>	output low hysteresis
<i>CMP_HYSTERESIS_M IDDLE</i>	output middle hysteresis
<i>CMP_HYSTERESIS_HI GH</i>	output high hysteresis
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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



## cmp\_noninverting\_input\_select

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

**Table 3-139. 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>	
cmp_periph	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
noninverting_input	noninverting input select
CMP_NONINVERTING_INPUT_PB0_PE9	CMP noninverting input PB0 for CMP0 or PE9 for CMP1
CMP_NONINVERTING_INPUT_PB2_PE12	CMP noninverting input PB2 for CMP0 or PE12 for CMP1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

## cmp\_output\_init

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

**Table 3-140. 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>	
cmp_periph	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
output_polarity	CMP output polarity
CMP_OUTPUT_POLARITY_INVERTED	output is inverted

<i>CMP_OUTPUT_POLARITY_NONINVERTED</i>	output is not inverted
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### cmp\_output\_mux\_config

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

**Table 3-141. Function cmp\_output\_mux\_config**

<b>Function name</b>	cmp_output_mux_config
<b>Function prototype</b>	void cmp_output_mux_config(cmp_enum cmp_periph, uint32_t cmp_output_sel);
<b>Function descriptions</b>	config comparator output port
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>cmp_output_sel</b>	CMP output selection
<i>CMP_AFSE_GPIO_PA6</i>	CMP alternate GPIO PA6
<i>CMP_AFSE_GPIO_PA8</i>	CMP alternate GPIO PA8
<i>CMP_AFSE_GPIO_PB12</i>	CMP alternate GPIO PB12
<i>CMP_AFSE_GPIO_PE6</i>	CMP alternate GPIO PE6
<i>CMP_AFSE_GPIO_PE15</i>	CMP alternate GPIO PE15
<i>CMP_AFSE_GPIO_PG2</i>	CMP alternate GPIO PG2
<i>CMP_AFSE_GPIO_PG3</i>	CMP alternate GPIO PG3
<i>CMP_AFSE_GPIO_PG4</i>	CMP alternate GPIO PG4
<i>CMP_AFSE_GPIO_PK0</i>	CMP alternate GPIO PK0

<i>CMP_AFSE_GPIO_PK</i> 1	CMP alternate GPIO PK1
<i>CMP_AFSE_GPIO_PK</i> 2	CMP alternate GPIO PK2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### cmp\_blanking\_init

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

**Table 3-142. Function cmp\_outputblank\_init**

<b>Function name</b>	cmp_blanking_init
<b>Function prototype</b>	void cmp_blanking_init(cmp_enum cmp_periph, uint32_t blanking_source_selection);
<b>Function descriptions</b>	CMP output blanking function init
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>blanking_source_selection</b>	blanking source selection
<i>CMP_BLANKING_NONE</i>	CMP no blanking source
<i>CMP_BLANKING_TIMER0_OC0</i>	CMP TIMER0_CH0 output compare signal selected as blanking source
<i>CMP_BLANKING_TIMER1_OC2</i>	CMP TIMER1_CH2 output compare signal selected as blanking source
<i>CMP_BLANKING_TIMER2_OC2</i>	CMP TIMER2_CH2 output compare signal selected as blanking source
<i>CMP_BLANKING_TIMER2_OC3</i>	CMP TIMER2_CH3 output compare signal selected as blanking source
<i>CMP_BLANKING_TIMER7_OC0</i>	CMP TIMER7_CH0 output compare signal selected as blanking source
<i>CMP_BLANKING_TIMER14_OC0</i>	CMP TIMER14_CH0 output compare signal selected as blanking source
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* initialize CMP0 blanking function */
```

```
cmp_blanking_init(CMP0, CMP_BLANKING_NONE);
```

### cmp\_enable

The description of cmp\_enable is shown as below:

**Table 3-143. Function cmp\_enable**

Function name	cmp_enable
Function prototype	void cmp_enable(cmp_enum cmp_periph);
Function descriptions	enable CMP
Precondition	-
The called functions	-
Input parameter{in}	
cmp_periph	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable CMP0 */
```

```
cmp_enable(CMP0);
```

### cmp\_disable

The description of cmp\_disable is shown as below:

**Table 3-144. 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 3-136. Enum cmp_enum</a>
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

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

### cmp\_window\_enable

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

**Table 3-145. Function cmp\_window\_enable**

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

Example:

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

### cmp\_window\_disable

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

**Table 3-146. Function cmp\_window\_disable**

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

Example:

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

```
cmp_window_disable();
```

### cmp\_lock\_enable

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

**Table 3-147. Function cmp\_lock\_enable**

<b>Function name</b>	cmp_lock_enable
<b>Function prototype</b>	void cmp_lock_enable(cmp_enum cmp_periph);
<b>Function descriptions</b>	lock the comparator
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
cmp_lock_enable(CMP0);
```

### cmp\_voltage\_scaler\_enable

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

**Table 3-148. Function cmp\_voltage\_scaler\_enable**

<b>Function name</b>	cmp_voltage_scaler_enable
<b>Function prototype</b>	void cmp_voltage_scaler_enable(cmp_enum cmp_periph);
<b>Function descriptions</b>	enable the voltage scaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CMP0 the voltage scaler */
```

```
cmp_voltage_scaler_enable(CMP0);
```

## cmp\_voltage\_scaler\_disable

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

**Table 3-149. Function cmp\_voltage\_scaler\_disable**

<b>Function name</b>	cmp_voltage_scaler_disable
<b>Function prototype</b>	void cmp_voltage_scaler_disable(cmp_enum cmp_periph);
<b>Function descriptions</b>	disable comparator the voltage scaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## cmp\_scaler\_bridge\_enable

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

**Table 3-150. Function cmp\_scaler\_bridge\_enable**

<b>Function name</b>	cmp_scaler_bridge_enable
<b>Function prototype</b>	void cmp_scaler_bridge_enable(cmp_enum cmp_periph);
<b>Function descriptions</b>	enable the scaler bridge
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## cmp\_scaler\_bridge\_disable

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

**Table 3-151. Function cmp\_scaler\_bridge\_disable**

<b>Function name</b>	cmp_scaler_bridge_disable
<b>Function prototype</b>	void cmp_scaler_bridge_disable(cmp_enum cmp_periph);
<b>Function descriptions</b>	disable the scaler bridge
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable CMP0 the scaler bridge */
```

```
cmp_scaler_bridge_disable(CMP0);
```

### cmp\_output\_level\_get

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

**Table 3-152. 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 3-136. Enum cmp_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the output level
<b>CMP_OUTPUTLEVEL_</b> <b>HIGH</b>	comparator output high
<b>CMP_OUTPUTLEVEL_</b> <b>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 3-153. 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 3-136. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	CMP interrupt flag
<b>CMP_FLAG_COMPAR E</b>	CMP compare flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

```
FlagStatus flag_value;
```

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

## cmp\_flag\_clear

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

**Table 3-154. Function cmp\_flag\_clear**

<b>Function name</b>	cmp_flag_clear
<b>Function prototype</b>	void cmp_flag_clear(cmp_enum cmp_periph, uint32_t flag);
<b>Function descriptions</b>	clear CMP flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	CMP interrupt flag
<b>CMP_FLAG_COMPAR E</b>	CMP compare flag
<b>Output parameter{out}</b>	
-	-

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 3-155. Function cmp\_interrupt\_enable**

<b>Function name</b>	cmp_interrupt_enable
<b>Function prototype</b>	void cmp_interrupt_enable(cmp_enum cmp_periph, uint32_t interrupt);
<b>Function descriptions</b>	enable CMP interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cmp_periph</b>	refer to enum <a href="#">Table 3-136. Enum cmp_enum</a>
<b>Input parameter{in}</b>	
<b>interrupt</b>	CMP interrupt
<b>CMP_INT_COMPARE</b>	CMP compare interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

### cmp\_interrupt\_disable

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

**Table 3-156. 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 3-136. 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 3-157. 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 3-136. 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>	
<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 3-158. 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 3-136. 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>	
-	-

Example:

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

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

## 3.5. CPDM

The Clock Phase Delay Module (CPDM) is used to delay the phase of the input clock and then output the clock. The CPDM registers are listed in chapter [3.5.1](#), the CPDM firmware functions are introduced in chapter [3.5.2](#).

### 3.5.1. Descriptions of Peripheral registers

CPDM registers are listed in the table shown as below:

**Table 3-159. CPDM Registers**

Registers	Descriptions
CPDM_CTL	CPDM control register
CPDM_CFG	CPDM configuration register

### 3.5.2. Descriptions of Peripheral functions

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

**Table 3-160. CPDM firmware function**

Function name	Function description
cpdm_enable	enable CPDM
cpdm_disable	disable CPDM
cpdm_delayline_sample_enable	enable CPDM delay line sample module
cpdm_delayline_sample_disable	disable CPDM delay line sample module

Function name	Function description
cpdm_output_clock_phase_select	select CPDM output clock phase
cpdm_delayline_length_valid_flag_get	get delay line length valid flag
cpdm_delayline_length_get	get delay line length
cpdm_clock_output	configure CPDM clock output

### Enum cpdm\_output\_phase\_enum

**Table 3-161. Enum cpdm\_output\_phase\_enum**

Member name	Function description
CPDM_OUTPUT_PHASE_SELECTION_0	output clock phase = input clock
CPDM_OUTPUT_PHASE_SELECTION_1	output clock phase = input clock + 1 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_2	output clock phase = input clock + 2 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_3	output clock phase = input clock + 3 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_4	output clock phase = input clock + 4 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_5	output clock phase = input clock + 5 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_6	output clock phase = input clock + 6 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_7	output clock phase = input clock + 7 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_8	output clock phase = input clock + 8 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_9	output clock phase = input clock + 9 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_10	output clock phase = input clock + 10 * UNIT delay
CPDM_OUTPUT_PHASE_SELECTION_11	output clock phase = input clock + 11 * UNIT delay

Member name	Function description
HASE_SELECTION_11	
CPDM_OUTPUT_P HASE_SELECTION_12	output clock phase = input clock + 12 * UNIT delay

## cpdm\_enable

The description of cpdm\_enable is shown as below:

**Table 3-162. Function cpdm\_enable**

<b>Function name</b>	cpdm_enable
<b>Function prototype</b>	void cpdm_enable(uint32_t cpdm_periph);
<b>Function descriptions</b>	enable CPDM
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cpdm_periph</b>	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CPDM */
cpdm_enable(CPDM_SDIO0);
```

## cpdm\_disable

The description of cpdm\_disable is shown as below:

**Table 3-163. Function cpdm\_disable**

<b>Function name</b>	cpdm_disable
<b>Function prototype</b>	void cpdm_disable(uint32_t cpdm_periph);
<b>Function descriptions</b>	disable CPDM
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cpdm_periph</b>	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable CPDM */
cpdm_disable(CPDM_SDIO0);
```

### cpdm\_delayline\_sample\_enable

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

**Table 3-164. Function cpdm\_delayline\_sample\_enable**

<b>Function name</b>	cpdm_delayline_sample_enable
<b>Function prototype</b>	void cpdm_delayline_sample_enable(uint32_t cpdm_periph);
<b>Function descriptions</b>	enable CPDM delay line sample module
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cpdm_periph</b>	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CPDM delay line sample module */
cpdm_delayline_sample_enable(CPDM_SDIO0);
```

### cpdm\_delayline\_sample\_disable

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

**Table 3-165. Function cpdm\_delayline\_sample\_disable**

<b>Function name</b>	cpdm_delayline_sample_disable
<b>Function prototype</b>	void cpdm_delayline_sample_disable(uint32_t cpdm_periph);
<b>Function descriptions</b>	disable CPDM delay line sample module
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cpdm_periph</b>	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable CPDM delay line sample module */
cpdm_delayline_sample_disable(CPDM_SDIO0);
```

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

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

**Table 3-166. Function cpdm\_output\_clock\_phase\_select**

<b>Function name</b>	cpdm_output_clock_phase_select
<b>Function prototype</b>	void cpdm_output_clock_phase_select(uint32_t cpdm_periph, cpdm_output_phase_enum output_clock_phase);
<b>Function descriptions</b>	select CPDM output clock phase
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cpdm_periph</b>	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
<b>Input parameter{in}</b>	
<b>output_clock_phase</b>	the output clock phase, refer to <a href="#">Table 3-161. Enum cpdm_output_phase_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select CPDM output clock phase */
cpdm_output_clock_phase_select(CPDM_SDIO0,  
CPDM_OUTPUT_PHASE_SELECTION_0);
```

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

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

**Table 3-167. Function cpdm\_delayline\_length\_valid\_flag\_get**

<b>Function name</b>	cpdm_delayline_length_valid_flag_get
<b>Function prototype</b>	FlagStatus cpdm_delayline_length_valid_flag_get(uint32_t cpdm_periph);
<b>Function descriptions</b>	get delay line length valid flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	



<b>cpdm_periph</b>	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

```
FlagStatus flag;
```

```
flag = cpdm_delayline_length_valid_flag_get(CPDM_SDIO0);
```

### cpdm\_delayline\_length\_get

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

**Table 3-168. Function cpdm\_delayline\_length\_get**

<b>Function name</b>	cpdm_delayline_length_get
<b>Function prototype</b>	uint16_t cpdm_delayline_length_get(uint32_t cpdm_periph);
<b>Function descriptions</b>	get delay line length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cpdm_periph</b>	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	0x00~0xFF

Example:

```
/* get delay line length */
```

```
uint16_t len;
```

```
len = cpdm_delayline_length_get(CPDM_SDIO0);
```

### cpdm\_clock\_output

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

**Table 3-169. Function cpdm\_clock\_output**

<b>Function name</b>	cpdm_clock_output
<b>Function prototype</b>	void cpdm_clock_output(uint32_t cpdm_periph, cpdm_output_phase_enum output_clock_phase);

<b>Function descriptions</b>	configure CPDM clock output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cpdm_periph</b>	the clock phase delay module of SDIO
<i>CPDM_SDIOx</i>	x = 0,1
<b>Input parameter{in}</b>	
<b>output_clock_phase</b>	the output clock phase, refer to <a href="#">Table 3-161. Enum cpdm_output_phase_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CPDM clock output */
cpdm_clock_output(CPDM_SDIO0, CPDM_OUTPUT_PHASE_SELECTION_1);
```

## 3.6. CRC

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. The CRC registers are listed in chapter [3.6.1](#), the CRC firmware functions are introduced in chapter [3.6.2](#).

### 3.6.1. Descriptions of Peripheral registers

CRC registers are listed in the table shown as below:

**Table 3-170. CRC Registers**

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

### 3.6.2. Descriptions of Peripheral functions

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

**Table 3-171. CRC firmware function**

Function name	Function description
crc_deinit	deinit CRC calculation unit

Function name	Function description
crc_init_data_register_write	write the initial value 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_reverse_output_data_disable	disable the reverse operation of output data
crc_reverse_output_data_enable	enable the reverse operation of output data
crc_input_data_reverse_config	configure the CRC input data function
crc_data_register_reset	reset data register to the value of initializaiton data register
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 3-172. Function crc\_deinit**

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

Example:

```
/* reset crc */
crc_deinit();
```

### crc\_init\_data\_register\_write

The description of crc\_init\_data\_register\_write is shown as below:

**Table 3-173. Function crc\_init\_data\_register\_write**

Function name	crc_init_data_register_write
Function prototype	void crc_init_data_register_write(uint32_t init_data)
Function descriptions	write the initializaiton data register
Precondition	-
The called functions	-

Input parameter{in}	
init_data	specify 32-bit data
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### crc\_data\_register\_read

The description of crc\_data\_register\_read is shown as below:

**Table 3-174. Function crc\_data\_register\_read**

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

Example:

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

### crc\_free\_data\_register\_read

The description of crc\_free\_data\_register\_read is shown as below:

**Table 3-175. Function crc\_free\_data\_register\_read**

Function name	crc_free_data_register_read
Function prototype	uint8_t crc_free_data_register_read(void);
Function descriptions	read the free data register
Precondition	-
The called functions	-
Input parameter{in}	

-	-
Output parameter{out}	
-	-
Return value	
uint8_t	8-bit value of the free data register (0-0xFF)

Example:

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

uint8_t crc_value = 0;

crc_value = crc_free_data_register_read();
```

### crc\_free\_data\_register\_write

The description of crc\_free\_data\_register\_write is shown as below:

**Table 3-176. Function crc\_free\_data\_register\_write**

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

Example:

```
/* write the free data register */

crc_free_data_register_write(0x11);
```

### crc\_reverse\_output\_data\_disable

The description of crc\_reverse\_output\_data\_disable is shown as below:

**Table 3-177. Function crc\_reverse\_output\_data\_disable**

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

Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
crc_reverse_output_data_disable();
```

### crc\_reverse\_output\_data\_enable

The description of crc\_reverse\_output\_data\_enable is shown as below:

**Table 3-178. Function crc\_reverse\_output\_data\_enable**

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

Example:

```
/* enable CRC reverse operation of output data */
```

```
crc_reverse_output_data_enable();
```

### crc\_input\_data\_reverse\_config

The description of crc\_input\_data\_reverse\_config is shown as below:

**Table 3-179. Function crc\_input\_data\_reverse\_config**

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

<i>CRC_INPUT_DATA_BYTE</i>	input data is reversed on 8 bits
<i>CRC_INPUT_DATA_HALFWORD</i>	input data is reversed on 16 bits
<i>CRC_INPUT_DATA_WORD</i>	input data is reversed on 32 bits
<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\_reset**

The description of `crc_data_register_reset` is shown as below:

**Table 3-180. Function `crc_data_register_reset`**

<b>Function name</b>	<code>crc_data_register_reset</code>
<b>Function prototype</b>	<code>void crc_data_register_reset(void);</code>
<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\_polynomial\_size\_set**

The description of `crc_polynomial_size_set` is shown as below:

**Table 3-181. Function `crc_polynomial_size_set`**

<b>Function name</b>	<code>crc_polynomial_size_set</code>
<b>Function prototype</b>	<code>void crc_polynomial_size_set(uint32_t poly_size)</code>
<b>Function descriptions</b>	configure the CRC size of polynomial function

<b>Precondition</b>	-
<b>The called functions</b>	-
<i>Input parameter{in}</i>	
<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 3-182. 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 3-183. Function `crc_single_data_calculate`**

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



<b>Function prototype</b>	uint32_t crc_single_data_calculate(uint32_t sdata, uint8_t data_format);
<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
<i>INPUT_FORMAT_WORD</i>	input data in word format
<i>INPUT_FORMAT_HALFWORD</i>	input data in half-word format
<i>INPUT_FORMAT_BYTE</i>	input data in byte format
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	CRC calculate value (0-0xFFFFFFFF)

Example:

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

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

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

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

### crc\_block\_data\_calculate

The description of crc\_block\_data\_calculate is shown as below:

**Table 3-184. Function crc\_block\_data\_calculate**

<b>Function name</b>	crc_block_data_calculate
<b>Function prototype</b>	uint32_t crc_block_data_calculate(void *array, uint32_t size, uint8_t data_format);
<b>Function descriptions</b>	CRC calculate a data array
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>array</b>	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
<i>INPUT_FORMAT_WORD</i>	input data in word format

<i>RD</i>	
<i>INPUT_FORMAT_HALFWORD</i>	input data in half-word format
<i>INPUT_FORMAT_BYTE</i>	input data in byte format
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	CRC calculate value (0-0xFFFFFFFF)

Example:

```

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

#define BUFFER_SIZE    6

uint32_t valcrc = 0;

static const uint32_t data_buffer[BUFFER_SIZE] = {

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

valcrc = crc_block_data_calculate(data_buffer, BUFFER_SIZE, INPUT_FORMAT_WORD);

```

## 3.7. CTC

The CTC unit trims the frequency of the IRC48M which is based on an external accurate reference signal source. It can adjust the calibration value to provide a precise IRC48M clock automatically or manually. The CTC registers are listed in chapter [3.7.1](#), the CTC firmware functions are introduced in chapter [3.7.2](#).

### 3.7.1. Descriptions of Peripheral registers

CTC registers are listed in the table shown as below:

**Table 3-185. CTC Registers**

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

### 3.7.2. Descriptions of Peripheral functions

CTC registers are listed in the table shown as below:

Table 3-186. CTC firmware function

Function name	Function description
ctc_deinit	reset CTC clock trim controller
ctc_counter_enable	enable CTC trim counter
ctc_counter_disable	disable CTC trim counter
ctc_irc48m_trim_value_config	configure the IRC48M trim value
ctc_software_refsource_pulse_generate	generate software reference source sync pulse
ctc_hardware_trim_mode_config	configure hardware automatically trim mode
ctc_refsource_polarity_config	configure reference signal source polarity
ctc_refsource_signal_select	select reference signal source
ctc_refsource_prescaler_config	configure reference signal source prescaler
ctc_clock_limit_value_config	configure clock trim base limit value
ctc_counter_reload_value_config	configure CTC counter reload value
ctc_counter_capture_value_read	read CTC counter capture value when reference sync pulse occurred
ctc_counter_direction_read	read CTC trim counter direction when reference sync pulse occurred
ctc_counter_reload_value_read	read CTC counter reload value
ctc_irc48m_trim_value_read	read the IRC48M trim value
ctc_flag_get	get CTC flag
ctc_flag_clear	clear CTC flag
ctc_interrupt_enable	enable the CTC interrupt
ctc_interrupt_disable	disable the CTC interrupt
ctc_interrupt_flag_get	get CTC interrupt flag
ctc_interrupt_flag_clear	clear CTC interrupt flag

### ctc\_deinit

The description of ctc\_deinit is shown as below:

Table 3-187. Function ctc\_deinit

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

Example:

```
/* reset CTC */
```

```
ctc_deinit();
```

### ctc\_counter\_enable

The description of ctc\_counter\_enable is shown as below:

**Table 3-188. Function ctc\_counter\_enable**

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

Example:

```
/* enable CTC trim counter */
```

```
ctc_counter_enable ();
```

### ctc\_counter\_disable

The description of ctc\_counter\_disable is shown as below:

**Table 3-189. Function ctc\_counter\_disable**

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

Example:

```
/* disable CTC trim counter */
```

```
ctc_counter_disable ();
```

### ctc\_irc48m\_trim\_value\_config

The description of ctc\_irc48m\_trim\_value\_config is shown as below:

**Table 3-190. Function ctc\_irc48m\_trim\_value\_config**

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

Example:

```
/* IRC48M trim value configuration */
ctc_irc48m_trim_value_config (0x01);
```

### ctc\_software\_refsource\_pulse\_generate

The description of ctc\_software\_refsource\_pulse\_generate is shown as below:

**Table 3-191. Function ctc\_software\_refsource\_pulse\_generate**

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

Example:

```
/* generate reference source sync pulse */
ctc_software_refsource_pulse_generate ();
```

### ctc\_hardware\_trim\_mode\_config

The description of ctc\_hardware\_trim\_mode\_config is shown as below:

**Table 3-192. Function ctc\_hardware\_trim\_mode\_config**

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

Example:

```
/* enable CTC hardware trim */
```

```
ctc_hardware_trim_mode_config (CTC_HARDWARE_TRIM_MODE_ENABLE);
```

### ctc\_refsource\_polarity\_config

The description of ctc\_refsource\_polarity\_config is shown as below:

**Table 3-193. Function ctc\_refsource\_polarity\_config**

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

Example:

```
/* set reference source polarity */
```

ctc\_refsource\_polarity\_config (CTC\_REFSOURCE\_POLARITY\_RISING);

### ctc\_refsource\_signal\_select

The description of ctc\_refsource\_signal\_select is shown as below:

**Table 3-194. Function ctc\_refsource\_signal\_select**

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

Example:

```
/* reference signal selection */
```

```
ctc_refsource_signal_select (CTC_REFSOURCE_LXTAL);
```

### ctc\_refsource\_prescaler\_config

The description of ctc\_refsource\_prescaler\_config is shown as below:

**Table 3-195. Function ctc\_refsource\_prescaler\_config**

<b>Function name</b>	ctc_refsource_prescaler_config
<b>Function prototype</b>	void ctc_refsource_prescaler_config(uint32_t prescaler);
<b>Function descriptions</b>	configure reference signal source prescaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>prescaler</b>	Prescaler factor
CTC_REFSOURCE_PSC_OFF	reference signal not divided
CTC_REFSOURCE_PSC_DIV2	reference signal divided by 2
CTC_REFSOURCE_PSC_DIV4	reference signal divided by 4

<i>CTC_REFSOURCE_P</i> <i>SC_DIV8</i>	reference signal divided by 8
<i>CTC_REFSOURCE_P</i> <i>SC_DIV16</i>	reference signal divided by 16
<i>CTC_REFSOURCE_P</i> <i>SC_DIV32</i>	reference signal divided by 32
<i>CTC_REFSOURCE_P</i> <i>SC_DIV64</i>	reference signal divided by 64
<i>CTC_REFSOURCE_P</i> <i>SC_DIV128</i>	reference signal divided by 128
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure reference signal source prescaler */
```

```
ctc_refsource_prescaler_config(CTC_REFSOURCE_PSC_DIV2);
```

### **ctc\_clock\_limit\_value\_config**

The description of `ctc_clock_limit_value_config` is shown as below:

**Table 3-196. Function `ctc_clock_limit_value_config`**

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

Example:

```
/* configure clock trim base limit value */
```

```
ctc_clock_limit_value_config (0x1F);
```

### **ctc\_counter\_reload\_value\_config**

The description of `ctc_counter_reload_value_config` is shown as below:



**Table 3-197. Function ctc\_counter\_reload\_value\_config**

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

Example:

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

### ctc\_counter\_capture\_value\_read

The description of ctc\_counter\_capture\_value\_read is shown as below:

**Table 3-198. Function ctc\_counter\_capture\_value\_read**

<b>Function name</b>	ctc_counter_capture_value_read
<b>Function prototype</b>	uint16_t ctc_counter_capture_value_read(void);
<b>Function descriptions</b>	read CTC counter capture value
<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 16-bit CTC counter capture value (0x0000 - 0xFFFF)

Example:

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

### ctc\_counter\_direction\_read

The description of ctc\_counter\_direction\_read is shown as below:

**Table 3-199. Function ctc\_counter\_direction\_read**

<b>Function name</b>	ctc_counter_direction_read
<b>Function prototype</b>	FlagStatus ctc_counter_direction_read(void);
<b>Function descriptions</b>	read CTC trim counter direction
<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 / RESET

Example:

```
/* read ctc counter direction */
```

```
FlagStatus ctc_direction = SET;
```

```
ctc_direction = ctc_counter_direction_read ();
```

### ctc\_counter\_reload\_value\_read

The description of ctc\_counter\_reload\_value\_read is shown as below:

**Table 3-200. Function ctc\_counter\_reload\_value\_read**

<b>Function name</b>	ctc_counter_reload_value_read
<b>Function prototype</b>	uint16_t ctc_counter_reload_value_read(void);
<b>Function descriptions</b>	read CTC counter reload value
<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>	Read 16-bit data of counter reload value (0x0000 - 0xFFFF)

Example:

```
/* read CTC counter reload value */
```

```
uint16_t ctc_reload_value = 0;
```

```
ctc_reload_value = ctc_counter_reload_value_read ();
```

### ctc\_irc48m\_trim\_value\_read

The description of ctc\_irc48m\_trim\_value\_read is shown as below:

Table 3-201. Function `ctc_irc48m_trim_value_read`

Function name	<code>ctc_irc48m_trim_value_read</code>
Function prototype	<code>uint8_t ctc_irc48m_trim_value_read(void);</code>
Function descriptions	read the IRC48M trim value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>uint8_t</code>	the 8-bit IRC48M trim value (0-63)

Example:

```
/* read the IRC48M trim value */
```

```
uint8_t ctc_trim_value = 0;
```

```
ctc_trim_value = ctc_irc48m_trim_value_read ();
```

### `ctc_flag_get`

The description of `ctc_flag_get` is shown as below:

Table 3-202. Function `ctc_flag_get`

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

Example:

```
/* get CTC flag status */
```

```
FlagStatus state = ctc_flag_get (CTC_FLAG_CKOK);
```

### ctc\_flag\_clear

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

**Table 3-203. Function ctc\_flag\_clear**

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

Example:

```
/* clear CTC flag status */
```

```
ctc_flag_clear (CTC_FLAG_CKOK);
```

### ctc\_interrupt\_enable

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

**Table 3-204. Function ctc\_interrupt\_enable**

<b>Function name</b>	ctc_interrupt_enable
<b>Function prototype</b>	void ctc_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable the CTC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	CTC interrupt
CTC_INT_CKOK	clock trim OK interrupt

<i>CTC_INT_CKWARN</i>	clock trim warning interrupt
<i>CTC_INT_ERR</i>	error interrupt
<i>CTC_INT_EREf</i>	expect reference interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable CTC clock trim OK interrupt */
ctc_interrupt_enable (CTC_INT_CKOK);
```

### ctc\_interrupt\_disable

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

**Table 3-205. Function ctc\_interrupt\_disable**

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

Example:

```
/* disable CTC clock trim OK interrupt */
ctc_interrupt_disable (CTC_INT_CKOK);
```

### ctc\_interrupt\_flag\_get

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

**Table 3-206. Function ctc\_interrupt\_flag\_get**

<b>Function name</b>	ctc_interrupt_flag_get
<b>Function prototype</b>	FlagStatus ctc_interrupt_flag_get(uint32_t int_flag);

<b>Function descriptions</b>	get CTC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	CTC interrupt flag
<i>CTC_INT_FLAG_CKOK</i>	clock trim OK interrupt
<i>CTC_INT_FLAG_CKWARN</i>	clock trim warning interrupt
<i>CTC_INT_FLAG_ERR</i>	error interrupt
<i>CTC_INT_FLAG_EREFP</i>	expect reference interrupt
<i>CTC_INT_FLAG_CKEERR</i>	clock trim error bit interrupt
<i>CTC_INT_FLAG_REFMISS</i>	reference sync pulse miss interrupt
<i>CTC_INT_FLAG_TRIMERR</i>	trim value error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get CTC interrupt flag status */
```

```
FlagStatus state = ctc_interrupt_flag_get (CTC_INT_FLAG_CKOK);
```

### ctc\_interrupt\_flag\_clear

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

**Table 3-207. Function ctc\_interrupt\_flag\_clear**

<b>Function name</b>	ctc_interrupt_flag_clear
<b>Function prototype</b>	void ctc_interrupt_flag_clear(uint32_t int_flag);
<b>Function descriptions</b>	clear CTC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	CTC interrupt flag
<i>CTC_INT_FLAG_CKOK</i>	clock trim OK interrupt
<i>CTC_INT_FLAG_CKWARN</i>	clock trim warning interrupt
<i>CTC_INT_FLAG_ERR</i>	error interrupt
<i>CTC_INT_FLAG_EREFP</i>	expect reference interrupt

<i>CTC_INT_FLAG_CKE</i> <i>RR</i>	clock trim error bit interrupt
<i>CTC_INT_FLAG_REF</i> <i>MISS</i>	reference sync pulse miss interrupt
<i>CTC_INT_FLAG_TRIM</i> <i>ERR</i>	trim value error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear CTC interrupt flag status */
ctc_interrupt_flag_clear (CTC_INT_FLAG_CKOK);
```

## 3.8. DAC

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

### 3.8.1. Peripheral register description

DAC registers are listed in the table shown as below:

**Table 3-208. DAC Registers**

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

Register	Descriptions
DAC_SKSTR0	DACx sample and keep sample time register 0
DAC_SKSTR1	DACx sample and keep sample time register 1
DAC_SKKTR	DACx sample and keep time register
DAC_SKRTR	DACx sample and keep refresh time register

### 3.8.2. Descriptions of Peripheral functions

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

**Table 3-209. DAC firmware functions**

Function name	Function description
dac_deinit	deinitialize DAC
dac_enable	enable DAC
dac_disable	disable DAC
dac_dma_enable	enable DAC DMA function
dac_dma_disable	disable DAC DMA function
dac_mode_config	configure DAC mode
dac_trimming_value_get	get the DACx trimming value
dac_trimming_value_set	set the DACx trimming value
dac_trimming_enable	enable the DACx trimming
dac_output_value_get	get DAC output value
dac_data_set	set DAC data holding register value
dac_trigger_enable	enable DAC trigger
dac_trigger_disable	disable DAC trigger
dac_trigger_source_config	configure DAC trigger source
dac_software_trigger_enable	enable DAC software trigger
dac_wave_mode_config	configure DAC wave mode
dac_lfsr_noise_config	configure DAC LFSR noise mode
dac_triangle_noise_config	configure DAC triangle noise mode
dac_concurrent_enable	enable DAC concurrent mode
dac_concurrent_disable	disable DAC concurrent mode
dac_concurrent_software_trigger_enable	enable DAC concurrent software trigger
dac_concurrent_data_set	set DAC concurrent mode data holding register value
dac_sample_keep_mode_config	set DAC sample and keep time value
dac_flag_get	get DAC flag
dac_flag_clear	clear DAC flag
dac_interrupt_enable	enable DAC interrupt
dac_interrupt_disable	disable DAC interrupt
dac_interrupt_flag_get	get DAC interrupt flag
dac_interrupt_flag_clear	clear DAC interrupt flag



## dac\_deinit

The description of dac\_deinit is shown as below:

**Table 3-210. Function dac\_deinit**

<b>Function name</b>	dac_deinit
<b>Function prototype</b>	void dac_deinit(uint32_t dac_periph);
<b>Function descriptions</b>	deinitialize DAC
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<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:

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

## dac\_enable

The description of dac\_enable is shown as below:

**Table 3-211. Function dac\_enable**

<b>Function name</b>	dac_enable
<b>Function prototype</b>	void dac_enable(uint32_t dac_periph, uint8_t dac_out);
<b>Function descriptions</b>	enable 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:

```
/* enable DAC0_OUT0 */
```

```
dac_enable(DAC0, DAC_OUT0);
```

### **dac\_disable**

The description of `dac_disable` is shown as below:

**Table 3-212. 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\_dma\_enable**

The description of `dac_dma_enable` is shown as below:

**Table 3-213. Function `dac_dma_enable`**

<b>Function name</b>	<code>dac_dma_enable</code>
<b>Function prototype</b>	<code>void dac_dma_enable(uint32_t dac_periph, uint8_t dac_out);</code>
<b>Function descriptions</b>	enable DAC DMA function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>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>	
-	-

Return value	
-	-

Example:

```
/* enable DAC0_OUT0 DMA function */
```

```
dac_dma_enable(DAC0, DAC_OUT0);
```

### **dac\_dma\_disable**

The description of `dac_dma_disable` is shown as below:

**Table 3-214. Function `dac_dma_disable`**

Function name	<code>dac_dma_disable</code>
Function prototype	<code>void dac_dma_disable(uint32_t dac_periph, uint8_t dac_out);</code>
Function descriptions	disable DAC DMA function
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 DMA function */
```

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

### **dac\_mode\_config**

The description of `dac_mode_config` is shown as below:

**Table 3-215. Function `dac_mode_config`**

Function name	<code>dac_mode_config</code>
Function prototype	<code>void dac_mode_config(uint32_t dac_periph, uint32_t dac_out, uint32_t mode);</code>
Function descriptions	configure DAC mode
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>Input parameter{in}</b>	
<b>mode</b>	DAC working mode
<i>NORMAL_PIN_BUFFON</i>	DAC_OUTx work in normal mode and connect to external pin with buffer enable
<i>NORMAL_PIN_PERIPHERAL_BUFFON</i>	DAC_OUTx work in normal mode and connect to external pin and on chip peripherals with buffer enable
<i>NORMAL_PIN_BUFFOFF</i>	DAC_OUTx work in normal mode and connect to external pin with buffer disable
<i>NORMAL_PIN_PERIPHERAL_BUFFOFF</i>	DAC_OUTx work in normal mode and connect to on chip peripherals with buffer disable
<i>SAMPLEKEEP_PIN_BUFFON</i>	DAC_OUTx work in sample and keep mode and connect to external pin with buffer enable
<i>SAMPLEKEEP_PIN_PERIPHERAL_BUFFON</i>	DAC_OUTx work in sample and keep mode and connect to external pin and on chip peripherals with buffer enable
<i>SAMPLEKEEP_PIN_BUFFOFF</i>	DAC_OUTx work in sample and keep mode and connect to external pin and on chip peripherals with buffer disable
<i>SAMPLEKEEP_PIN_PERIPHERAL_BUFFOFF</i>	DAC_OUTx work in sample and keep mode and connect to on chip peripherals with buffer disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

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

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

The description of `dac_trimming_value_get` is shown as below:

**Table 3-216. Function `dac_trimming_value_get`**

<b>Function name</b>	<code>dac_trimming_value_get</code>
<b>Function prototype</b>	<code>void dac_trimming_value_get(uint32_t dac_periph, uint32_t dac_out,);</code>
<b>Function descriptions</b>	get the DACx trimming value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral

<i>DACx</i>	DAC peripheral selection(x = 0)
<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>Uint32_t</b>	DACx trimming value

Example:

```
/* get the DAC0_OUT0 trimming value */
data = dac_trimming_value_get (DAC0, DAC_OUT0);
```

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

The description of `dac_trimming_value_set` is shown as below:

**Table 3-217. Function `dac_trimming_value_set`**

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

Example:

```
/* set the DAC0_OUT0 trimming value */
dac_trimming_value_set(DAC0, DAC_OUT0, 1);
```

### **dac\_trimming\_enable**

The description of `dac_trimming_enable` is shown as below:

Table 3-218. Function `dac_trimming_enable`

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

Example:

```
/* enable the DAC0_OUT0 trimming */
dac_trimming_enable (DAC0, DAC_OUT0);
```

### `dac_output_value_get`

The description of `dac_output_value_get` is shown as below:

Table 3-219. 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 3-220. Function `dac_data_set`**

<b>Function name</b>	<code>dac_data_set</code>
<b>Function prototype</b>	<code>void dac_data_set(uint32_t dac_periph, uint8_t dac_out, uint32_t dac_align, uint16_t data);</code>
<b>Function descriptions</b>	set DAC data holding register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
<b>Input parameter{in}</b>	
<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Input parameter{in}</b>	
<b>dac_align</b>	DAC data alignment mode
<i>DAC_ALIGN_12B_R</i>	12-bit right-aligned data
<i>DAC_ALIGN_12B_L</i>	12-bit left-aligned data
<i>DAC_ALIGN_8B_R</i>	8-bit right-aligned data
<b>Input parameter{in}</b>	
<b>data</b>	data to be loaded (0~4095)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set DAC0_OUT0 data holding register value */

dac_data_set(DAC0, DAC_OUT0, DAC_ALIGN_8B_R, 0xFF);
```

### **dac\_trigger\_enable**

The description of `dac_trigger_enable` is shown as below:

**Table 3-221. Function `dac_trigger_enable`**

<b>Function name</b>	<code>dac_trigger_enable</code>
<b>Function prototype</b>	<code>void dac_trigger_enable(uint32_t dac_periph, uint8_t dac_out);</code>
<b>Function descriptions</b>	enable DAC trigger

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

Example:

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

### **dac\_trigger\_disable**

The description of dac\_trigger\_disable is shown as below:

**Table 3-222. Function dac\_trigger\_disable**

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

Example:

```
/* disable DAC0_OUT0 trigger */
dac_trigger_disable(DAC0, DAC_OUT0);
```



## dac\_trigger\_source\_config

The description of dac\_trigger\_source\_config is shown as below:

**Table 3-223. Function dac\_trigger\_source\_config**

<b>Function name</b>	dac_trigger_source_config
<b>Function prototype</b>	void dac_trigger_source_config(uint32_t dac_periph, uint8_t dac_out, uint32_t triggersource);
<b>Function descriptions</b>	configure DAC trigger source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection (x = 0)
<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 TRIGSEL
<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_T1_TRGO);
```

## dac\_software\_trigger\_enable

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

**Table 3-224. Function dac\_software\_trigger\_enable**

<b>Function name</b>	dac_software_trigger_enable
<b>Function prototype</b>	void dac_software_trigger_enable(uint32_t dac_periph, uint8_t dac_out);
<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 3-225. 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 3-226. Function `dac_lfsr_noise_config`**

<b>Function name</b>	<code>dac_lfsr_noise_config</code>
<b>Function prototype</b>	<code>void dac_lfsr_noise_config(uint32_t dac_periph, uint8_t dac_out, uint32_t unmask_bits);</code>
<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 3-227. 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_AMPLITUDE_x</i>	$x = 2^n - 1$ (n = 1..12)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DAC0_OUT0 triangle noise mode */
```

```
dac_triangle_noise_config(DAC0, DAC_OUT0, DAC_TRIANGLE_AMPLITUDE_1);
```

### **dac\_concurrent\_enable**

The description of `dac_concurrent_enable` is shown as below:

**Table 3-228. 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 3-229. Function dac\_concurrent\_disable**

<b>Function name</b>	dac_concurrent_disable
<b>Function prototype</b>	void dac_concurrent_disable(uint32_t dac_periph);
<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>	
-	-
<b>Return value</b>	
-	-

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 3-230. Function dac\_concurrent\_software\_trigger\_enable**

<b>Function name</b>	dac_concurrent_software_trigger_enable
<b>Function prototype</b>	void dac_concurrent_software_trigger_enable(uint32_t dac_periph);
<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 3-231. Function `dac_concurrent_data_set`

Function name	<code>dac_concurrent_data_set</code>
Function prototype	<code>void dac_concurrent_data_set(uint32_t dac_periph, uint32_t dac_align, uint16_t data0, uint16_t data1);</code>
Function descriptions	set DAC concurrent mode 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_align</code>	DAC data alignment mode
<code>DAC_ALIGN_8B_R</code>	8-bit right-aligned data
<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>data0</code>	DACx_OUT0 data to be loaded (0~4095)
Input parameter{in}	
<code>data1</code>	DACx_OUT1 data to be loaded (0~4095)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set DAC0 concurrent mode data holding register value */
```

```
dac_concurrent_data_set(DAC0, DAC_ALIGN_8B_R, 0xFF, 0xFF);
```

### `dac_sample_keep_mode_config`

The description of `dac_sample_keep_mode_config` is shown as below:

Table 3-232. Function `dac_sample_keep_mode_config`

Function name	<code>dac_sample_keep_mode_config</code>
Function prototype	<code>void dac_sample_keep_mode_config(uint32_t dac_periph, uint32_t dac_out, uint32_t sample_time, uint32_t keep_time, uint32_t refresh_time);</code>
Function descriptions	set DAC sample and keep time 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}	

<b>dac_out</b>	DAC output
<i>DAC_OUTx</i>	DAC output channel selection(x = 0,1)
<b>Input parameter{in}</b>	
<b>sample_time</b>	DAC sample time
<b>Input parameter{in}</b>	
<b>keep_time</b>	DAC keep time
<b>Input parameter{in}</b>	
<b>refresh_time</b>	DAC refresh time
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set DAC0_OUT0 sample and keep time value */
```

```
dac_sample_keep_mode_config (DAC0, DAC_OUT0, 1, 1, 1);
```

### **dac\_flag\_get**

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

**Table 3-233. Function dac\_flag\_get**

<b>Function name</b>	dac_flag_get
<b>Function prototype</b>	FlagStatus dac_flag_get(uint32_t dac_periph, uint32_t flag);
<b>Function descriptions</b>	get DAC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
<b>Input parameter{in}</b>	
<b>flag</b>	the DAC status flags
<i>DAC_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun flag
<i>DAC_FLAG_CALF0</i>	DACx_OUT0 calibration offset flag
<i>DAC_FLAG_BWT0</i>	DACx_OUT0 sample and keep write enable flag
<i>DAC_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun flag
<i>DAC_FLAG_CALF1</i>	DACx_OUT1 calibration offset flag
<i>DAC_FLAG_BWT1</i>	DACx_OUT1 sample and keep write enable flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	the state of DAC bit (SET or RESET)

Example:

```
/* get DAC0 flag */
```

```
FlagStatus flag;
```

```
flag = dac_flag_get(DAC0, DAC_FLAG_DDUDR0);
```

### **dac\_flag\_clear**

The description of `dac_flag_clear` is shown as below:

**Table 3-234. Function `dac_flag_clear`**

<b>Function name</b>	<code>dac_flag_clear</code>
<b>Function prototype</b>	<code>void dac_flag_clear(uint32_t dac_periph, uint32_t flag);</code>
<b>Function descriptions</b>	clear DAC flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
<b>Input parameter{in}</b>	
<b>flag</b>	DAC flag
<i>DAC_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun flag
<i>DAC_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear DAC0 flag */
```

```
dac_flag_clear(DAC0, DAC_FLAG_DDUDR0);
```

### **dac\_interrupt\_enable**

The description of `dac_interrupt_enable` is shown as below:

**Table 3-235. Function `dac_interrupt_enable`**

<b>Function name</b>	<code>dac_interrupt_enable</code>
<b>Function prototype</b>	<code>void dac_interrupt_enable(uint32_t dac_periph, uint32_t interrupt);</code>
<b>Function descriptions</b>	enable DAC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
<b>Input parameter{in}</b>	



<b>interrupt</b>	the DAC interrupt
<i>DAC_INT_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt
<i>DAC_INT_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DAC0 interrupt */
```

```
dac_interrupt_enable (DAC0, DAC_INT_DDUDRIE0);
```

### **dac\_interrupt\_disable**

The description of `dac_interrupt_disable` is shown as below:

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

<b>Function name</b>	<code>dac_interrupt_disable</code>
<b>Function prototype</b>	<code>void dac_interrupt_disable(uint32_t dac_periph, uint32_t interrupt);</code>
<b>Function descriptions</b>	disable DAC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
<b>Input parameter{in}</b>	
<b>interrupt</b>	the DAC interrupt
<i>DAC_INT_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt
<i>DAC_INT_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DAC0 interrupt */
```

```
dac_interrupt_disable (DAC0, DAC_INT_DDUDRIE0);
```

### **dac\_interrupt\_flag\_get**

The description of `dac_interrupt_flag_get` is shown as below:

**Table 3-237. Function `dac_interrupt_flag_get`**

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

<b>Function prototype</b>	FlagStatus dac_interrupt_flag_get(uint32_t dac_periph, uint32_t int_flag);
<b>Function descriptions</b>	get DAC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
<b>Input parameter{in}</b>	
<b>int_flag</b>	DAC interrupt flag
<i>DAC_INT_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt flag
<i>DAC_INT_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	the state of DAC interrupt flag(SET or RESET)

Example:

```
/* get DAC0 interrupt flag */
```

```
flag = dac_interrupt_flag_get(DAC0, DAC_INT_FLAG_DDUDR0);
```

### **dac\_interrupt\_flag\_clear**

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

**Table 3-238. Function dac\_interrupt\_flag\_clear**

<b>Function name</b>	dac_interrupt_flag_clear
<b>Function prototype</b>	Void dac_interrupt_flag_clear(uint32_t dac_periph, uint32_t int_flag);
<b>Function descriptions</b>	clear DAC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dac_periph</b>	DAC peripheral
<i>DACx</i>	DAC peripheral selection(x = 0)
<b>Input parameter{in}</b>	
<b>int_flag</b>	DAC interrupt flag
<i>DAC_INT_FLAG_DDUDR0</i>	DACx_OUT0 DMA underrun interrupt flag
<i>DAC_INT_FLAG_DDUDR1</i>	DACx_OUT1 DMA underrun interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear DAC0 interrupt flag */
```

```
dac_interrupt_flag_clear(DAC0, DAC_INT_FLAG_DDUDR0);
```

## 3.9. DBG

The DBG hold unit helps debugger to debug power saving mode. The DBG registers are listed in chapter [3.9.1](#). the DBG firmware functions are introduced in chapter [3.9.2](#).

### 3.9.1. Descriptions of Peripheral registers

DBG registers are listed in the table shown as below:

**Table 3-239. DBG Registers**

Registers	Descriptions
DBG_ID	DBG ID code register
DBG_CTL0	DBG control register 0
DBG_CTL1	DBG control register 1
DBG_CTL2	DBG control register 2
DBG_CTL3	DBG control register 3
DBG_CTL4	DBG control register 4

### 3.9.2. Descriptions of Peripheral functions

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

**Table 3-240. DBG firmware function**

Function name	Function description
dbg_deinit	deinitialize the DBG
dbg_id_get	read DBG_ID code register
dbg_low_power_enable	enable low power behavior when the MCU is in debug mode
dbg_low_power_disable	disable low power behavior when the MCU is in debug mode
dbg_periph_enable	enable peripheral behavior when the MCU is in debug mode
dbg_periph_disable	disable peripheral behavior when the MCU is in debug mode
dbg_trace_pin_enable	enable trace pin assignment
dbg_trace_pin_disable	disable trace pin assignment
dbg_trace_pin_mode_set	set trace pin mode

### Enum dbg\_periph\_enum

**Table 3-241. Enum dbg\_periph\_enum**

Member name	Function description
DBG_FWDGT_HOLD	debug FWDGT kept when core is halted
DBG_WWDGT_HOLD	debug WWDGT kept when core is halted
DBG_TIMERx_HOLD	hold TIMEx (x=0,1,2,3,4,5,6,7,14,15,16,22,23,30,31,40,41,42,43,44,50,51) counter when core is halted
DBG_I2Cx_HOLD	hold I2Cx (x=0,1,2,3) smbus when core is halted

Member name	Function description
DBG_CANx_HOLD	hold CANx (x=0,1,2) counter when core is halted
DBG_RTC_HOLD	hold RTC calendar and wakeup counter when core is halted

## dbg\_deinit

The description of dbg\_deinit is shown as below:

**Table 3-242. Function dbg\_deinit**

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

Example:

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

```
dbg_deinit();
```

## dbg\_id\_get

The description of dbg\_id\_get is shown as below:

**Table 3-243. Function dbg\_id\_get**

Function name	dbg_id_get
Function prototype	uint32_t dbg_id_get(void);
Function descriptions	read DBG_ID code register
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uint32_t	DBG_ID code (0-0xFFFFFFFF)

Example:

```
/* read DBG_ID code register */
```

```
uint32_t id_value = 0;

id_value = dbg_id_get();
```

### dbg\_low\_power\_enable

The description of dbg\_low\_power\_enable is shown as below:

**Table 3-244. 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 3-245. Function dbg\_low\_power\_disable**

<b>Function name</b>	dbg_low_power_disable
<b>Function prototype</b>	void dbg_low_power_disable(uint32_t dbg_low_power);
<b>Function descriptions</b>	disable low power behavior when the mcu is in debug mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dbg_low_power</b>	low power mode
<i>DBG_LOW_POWER_SLEEP</i>	do not keep debugger connection during sleep mode

<i>DBG_LOW_POWER_D EEPSLEEP</i>	do not keep debugger connection during deepsleep mode
<i>DBG_LOW_POWER_S TANDBY</i>	do not keep debugger connection during standby mode
<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 3-246. Function dbg\_periph\_enable**

<b>Function name</b>	dbg_periph_enable
<b>Function prototype</b>	void dbg_periph_enable(dbg_periph_enum dbg_periph);
<b>Function descriptions</b>	Enable peripheral behavior when the mcu is in debug mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dbg_periph</b>	Peripheral refer to <a href="#">Table 3-241. Enum dbg_periph_enum</a>
<i>DBG_FWDGT_HOLD</i>	debug FWDGT kept when core is halted
<i>DBG_WWDGT_HOLD</i>	debug WWDGT kept when core is halted
<i>DBG_TIMERx_HOLD</i>	hold TIMEx (x=0,1,2,3,4,5,6,7,14,15,16,22,23,30,31,40,41,42,43,44,50,51)counter when core is halted
<i>DBG_I2Cx_HOLD</i>	hold I2Cx (x=0,1,2,3) smbus when core is halted
<i>DBG_CANx_HOLD</i>	hold CANx (x=0,1,2) counter when core is halted
<i>DBG_RTC_HOLD</i>	hold RTC calendar and wakeup counter when core is halted
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
dbg_periph_enable(DBG_TIMER1_HOLD);
```

## dbg\_periph\_disable

The description of dbg\_periph\_disable is shown as below:

**Table 3-247. 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 3-241. Enum dbg_periph_enum</a>
<i>DBG_FWDGT_HOLD</i>	debug FWDGT kept when core is halted
<i>DBG_WWDGT_HOLD</i>	debug WWDGT kept when core is halted
<i>DBG_TIMERx_HOLD</i>	hold TIMERx (x=0,1,2,3,4,5,6,7,14,15,16,22,23,30,31,40,41,42,43,44,50,51)counter when core is halted
<i>DBG_I2Cx_HOLD</i>	hold I2Cx (x=0,1,2,3) smbus when core is halted
<i>DBG_CANx_HOLD</i>	hold CANx (x=0,1,2) counter when core is halted
<i>DBG_RTC_HOLD</i>	hold RTC calendar and wakeup counter when core is halted
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
dbg_periph_disable(DBG_TIMER1_HOLD);
```

## dbg\_trace\_pin\_enable

The description of dbg\_trace\_pin\_enable is shown as below:

**Table 3-248. Function dbg\_trace\_pin\_enable**

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



-	-
---	---

Example:

```
/* enable trace pin assignment */
dbg_trace_pin_enable();
```

### dbg\_trace\_pin\_disable

The description of dbg\_trace\_pin\_disable is shown as below:

**Table 3-249. Function dbg\_trace\_pin\_disable**

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

Example:

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

### dbg\_trace\_pin\_mode\_set

The description of dbg\_trace\_pin\_mode\_set is shown as below:

**Table 3-250. Function dbg\_trace\_pin\_mode\_set**

<b>Function name</b>	dbg_trace_pin_mode_set
<b>Function prototype</b>	void dbg_trace_pin_mode_set(uint32_t trace_mode);
<b>Function descriptions</b>	Trace pin mode selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>trace_mode</b>	trace pin mode selection
TRACE_MODE_ASYNC	trace pin used for async mode
TRACE_MODE_SYNC_DATASIZE_1	trace pin used for sync mode and data size is 1
TRACE_MODE_SYNC	trace pin used for sync mode and data size is 2

<code>_DATASIZE_2</code>	
<code>TRACE_MODE_SYNC</code> <code>_DATASIZE_4</code>	trace pin used for sync mode and data size is 4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
dbg_trace_pin_mode_set(TRACE_MODE_ASYNC);
```

## 3.10. DMA / DMAMUX

The direct memory access (DMA) controller provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. The DMA registers are listed in chapter [3.10.1](#), the DMA firmware functions are introduced in chapter [3.10.2](#).

DMAMUX is a transmission scheduler for DMA requests. The DMAMUX request multiplexer is used for routing a DMA request line between the peripherals / generated DMA request (from the DMAMUX request generator) and the DMA controller. The DMAMUX registers are listed in chapter [3.10.1](#), the DMAMUX firmware functions are introduced in chapter [3.10.2](#).

### 3.10.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

**Table 3-251. DMA Registers**

Registers	Descriptions
DMA_INTF0	Interrupt flag register 0
DMA_INTF1	Interrupt flag register 1
DMA_INTC0	Interrupt flag clear register 0
DMA_INTC1	Interrupt flag clear register 1
DMA_CHxCTL (x=0..7)	Channel x control register
DMA_CHxCNT (x=0..7)	Channel x counter register
DMA_CHxPADDR (x=0..7)	Channel x peripheral base address register
DMA_CHxM0ADDR (x=0..7)	Channel x memory 0 base address register

Registers	Descriptions
DMA_CHxM1ADDR (x=0..7)	Channel x memory 1 base address register
DMA_CHxFCTL (x=0..7)	Channel x FIFO control register

DMAMUX registers are listed in the table shown as below:

**Table 3-252. DMAMUX Registers**

Registers	Descriptions
DMAMUX_RM_CHx CFG (x=0..15)	Request multiplexer channel x configuration register
DMAMUX_RM_INT F	Request multiplexer channel interrupt flag register
DMAMUX_RM_INT C	Request multiplexer channel interrupt flag clear register
DMAMUX_RG_CHx CFG (x=0..7)	Request generator channel x configuration register
DMAMUX_RG_INT F	Request generator channel interrupt flag register
DMAMUX_RG_INT C	Rquest generator channel interrupt flag clear register

### 3.10.2. Descriptions of Peripheral functions

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

**Table 3-253. DMA firmware function**

Function name	Function description
dma_deinit	deinitialize DMA registers of a channel
dma_single_data_para_struct_init	initialize the DMA single data mode parameters struct with the default values
dma_multi_data_para_struct_init	initialize the DMA multi data mode parameters struct with the default values
dma_single_data_mode_init	initialize DMA single data mode
dma_multi_data_mode_init	initialize DMA multi data mode
dma_periph_address_config	configure DMA peripheral base address
dma_memory_address_config	configure DMA memory base address
dma_transfer_number_config	configure 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

Function name	Function description
dma_memory_burst_beats_config	configure transfer burst beats of memory
dma_periph_burst_beats_config	configure transfer burst beats of peripheral
dma_memory_width_config	configure transfer data size of memory
dma_periph_width_config	configure transfer data size of peripheral
dma_memory_address_generation_config	configure memory address generation algorithm
dma_peripheral_address_generation_config	configure peripheral address generation algorithm
dma_circulation_enable	enable DMA circulation mode
dma_circulation_disable	disable DMA circulation mode
dma_channel_enable	enable DMA channel
dma_channel_disable	disable DMA channel
dma_transfer_direction_config	configure the direction of data transfer on the channel
dma_switch_buffer_mode_config	configure DMA switch buffer mode
dma_using_memory_get	get DMA using memory
dma_switch_buffer_mode_enable	enable DMA switch buffer mode
dma_switch_buffer_mode_disable	disable DMA switch buffer mode
dma_fifo_status_get	get DMA FIFO status
dma_flag_get	get DMA flag
dma_flag_clear	clear DMA flag
dma_interrupt_enable	enable DMA interrupt
dma_interrupt_disable	disable DMA interrupt
dma_interrupt_flag_get	get DMA interrupt flag
dma_interrupt_flag_clear	clear DMA interrupt flag

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

**Table 3-254. 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

Function name	Function description
disable	
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\_multi\_data\_parameter\_struct

**Table 3-255. Structure dma\_multi\_data\_parameter\_struct**

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

### Structure dma\_single\_data\_parameter\_struct

**Table 3-256. Structure dma\_single\_data\_parameter\_struct**

Member name	Function description
request	channel input identification

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

### Structure dmamux\_sync\_parameter\_struct

**Table 3-257. 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 3-258. Structure dmamux\_gen\_parameter\_struct**

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

### Enum dma\_channel\_enum

**Table 3-259. Enum dma\_channel\_enum**

Member name	Function description
DMA_CH0	DMA channel 0
DMA_CH1	DMA channel 1
DMA_CH2	DMA channel 2
DMA_CH3	DMA channel 3
DMA_CH4	DMA channel 4
DMA_CH5	DMA channel 5
DMA_CH6	DMA channel 6
DMA_CH7	DMA channel 7

## Enum dmamux\_multiplexer\_channel\_enum

**Table 3-260. Enum dmamux\_multiplexer\_channel\_enum**

Member name	Function description
DMAMUX_MUXCH 0	DMAMUX request multiplexer channel 0
DMAMUX_MUXCH 1	DMAMUX request multiplexer channel 1
DMAMUX_MUXCH 2	DMAMUX request multiplexer channel 2
DMAMUX_MUXCH 3	DMAMUX request multiplexer channel 3
DMAMUX_MUXCH 4	DMAMUX request multiplexer channel 4
DMAMUX_MUXCH 5	DMAMUX request multiplexer channel 5
DMAMUX_MUXCH 6	DMAMUX request multiplexer channel 6
DMAMUX_MUXCH 7	DMAMUX request multiplexer channel 7
DMAMUX_MUXCH 8	DMAMUX request multiplexer channel 8
DMAMUX_MUXCH 9	DMAMUX request multiplexer channel 9
DMAMUX_MUXCH 10	DMAMUX request multiplexer channel 10
DMAMUX_MUXCH 11	DMAMUX request multiplexer channel 11
DMAMUX_MUXCH 12	DMAMUX request multiplexer channel 12
DMAMUX_MUXCH 13	DMAMUX request multiplexer channel 13
DMAMUX_MUXCH 14	DMAMUX request multiplexer channel 14
DMAMUX_MUXCH 15	DMAMUX request multiplexer channel 15

## Enum dmamux\_generator\_channel\_enum

**Table 3-261. 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

Member name	Function description
DMAMUX_GENCH3	DMAMUX request generator channel3
DMAMUX_GENCH4	DMAMUX request generator channel4
DMAMUX_GENCH5	DMAMUX request generator channel5
DMAMUX_GENCH6	DMAMUX request generator channel6
DMAMUX_GENCH7	DMAMUX request generator channel7

## Enum dmamux\_interrupt\_enum

**Table 3-262. Enum dmamux\_interrupt\_enum**

Member name	Function description
DMAMUX_INT_MU XCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun interrupt
DMAMUX_INT_MU XCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun interrupt
DMAMUX_INT_MU XCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun interrupt
DMAMUX_INT_MU XCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun interrupt
DMAMUX_INT_MU XCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun interrupt
DMAMUX_INT_MU XCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun interrupt
DMAMUX_INT_MU XCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun interrupt
DMAMUX_INT_MU XCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun interrupt
DMAMUX_INT_MU XCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun interrupt
DMAMUX_INT_MU XCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun interrupt
DMAMUX_INT_MU XCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun interrupt
DMAMUX_INT_MU XCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun interrupt
DMAMUX_INT_MU XCH12_SO	DMAMUX request multiplexer channel 12 synchronization overrun interrupt
DMAMUX_INT_MU XCH13_SO	DMAMUX request multiplexer channel 13 synchronization overrun interrupt
DMAMUX_INT_MU XCH14_SO	DMAMUX request multiplexer channel 14 synchronization overrun interrupt
DMAMUX_INT_MU XCH15_SO	DMAMUX request multiplexer channel 15 synchronization overrun interrupt
DMAMUX_INT_GE	DMAMUX request generator channel 0 trigger overrun interrupt



NCH0_TO	
DMAMUX_INT_GE NCH1_TO	DMAMUX request generator channel 1 trigger overrun interrupt
DMAMUX_INT_GE NCH2_TO	DMAMUX request generator channel 2 trigger overrun interrupt
DMAMUX_INT_GE NCH3_TO	DMAMUX request generator channel 3 trigger overrun interrupt
DMAMUX_INT_GE NCH4_TO	DMAMUX request generator channel 4 trigger overrun interrupt
DMAMUX_INT_GE NCH5_TO	DMAMUX request generator channel 5 trigger overrun interrupt
DMAMUX_INT_GE NCH6_TO	DMAMUX request generator channel 6 trigger overrun interrupt
DMAMUX_INT_GE NCH7_TO	DMAMUX request generator channel 7 trigger overrun interrupt

### Enum dmamux\_flag\_enum

**Table 3-263. Enum dmamux\_flag\_enum**

Member name	Function description
DMAMUX_FLAG_M UXCH0_SO	DMAMUX request multiplexer channel 0 synchronization overrun flag
DMAMUX_FLAG_M UXCH1_SO	DMAMUX request multiplexer channel 1 synchronization overrun flag
DMAMUX_FLAG_M UXCH2_SO	DMAMUX request multiplexer channel 2 synchronization overrun flag
DMAMUX_FLAG_M UXCH3_SO	DMAMUX request multiplexer channel 3 synchronization overrun flag
DMAMUX_FLAG_M UXCH4_SO	DMAMUX request multiplexer channel 4 synchronization overrun flag
DMAMUX_FLAG_M UXCH5_SO	DMAMUX request multiplexer channel 5 synchronization overrun flag
DMAMUX_FLAG_M UXCH6_SO	DMAMUX request multiplexer channel 6 synchronization overrun flag
DMAMUX_FLAG_M UXCH7_SO	DMAMUX request multiplexer channel 7 synchronization overrun flag
DMAMUX_FLAG_M UXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun flag
DMAMUX_FLAG_M UXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun flag
DMAMUX_FLAG_M UXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun flag
DMAMUX_FLAG_M UXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun flag

DMAMUX_FLAG_MUXCH12_SO	DMAMUX request multiplexer channel 12 synchronization overrun flag
DMAMUX_FLAG_MUXCH13_SO	DMAMUX request multiplexer channel 13 synchronization overrun flag
DMAMUX_FLAG_MUXCH14_SO	DMAMUX request multiplexer channel 14 synchronization overrun flag
DMAMUX_FLAG_MUXCH15_SO	DMAMUX request multiplexer channel 15 synchronization overrun flag
DMAMUX_FLAG_GENERCH0_TO	DMAMUX request generator channel 0 trigger overrun flag
DMAMUX_FLAG_GENERCH1_TO	DMAMUX request generator channel 1 trigger overrun flag
DMAMUX_FLAG_GENERCH2_TO	DMAMUX request generator channel 2 trigger overrun flag
DMAMUX_FLAG_GENERCH3_TO	DMAMUX request generator channel 3 trigger overrun flag
DMAMUX_FLAG_GENERCH4_TO	DMAMUX request generator channel 4 trigger overrun flag
DMAMUX_FLAG_GENERCH5_TO	DMAMUX request generator channel 5 trigger overrun flag
DMAMUX_FLAG_GENERCH6_TO	DMAMUX request generator channel 6 trigger overrun flag
DMAMUX_FLAG_GENERCH7_TO	DMAMUX request generator channel 7 trigger overrun flag

### Enum dmamux\_interrupt\_flag\_enum

**Table 3-264. 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

Member name	Function description
G_MUXCH7_SO	
DMAMUX_INT_FLG G_MUXCH8_SO	DMAMUX request multiplexer channel 8 synchronization overrun interrupt flag
DMAMUX_INT_FLG G_MUXCH9_SO	DMAMUX request multiplexer channel 9 synchronization overrun interrupt flag
DMAMUX_INT_FLG G_MUXCH10_SO	DMAMUX request multiplexer channel 10 synchronization overrun interrupt flag
DMAMUX_INT_FLG G_MUXCH11_SO	DMAMUX request multiplexer channel 11 synchronization overrun interrupt flag
DMAMUX_INT_FLG G_MUXCH12_SO	DMAMUX request multiplexer channel 12 synchronization overrun interrupt flag
DMAMUX_INT_FLG G_MUXCH13_SO	DMAMUX request multiplexer channel 13 synchronization overrun interrupt flag
DMAMUX_INT_FLG G_MUXCH14_SO	DMAMUX request multiplexer channel 14 synchronization overrun interrupt flag
DMAMUX_INT_FLG G_MUXCH15_SO	DMAMUX request multiplexer channel 15 synchronization overrun interrupt flag
DMAMUX_INT_FLG G_GENCH0_TO	DMAMUX request generator channel 0 trigger overrun interrupt flag
DMAMUX_INT_FLG G_GENCH1_TO	DMAMUX request generator channel 1 trigger overrun interrupt flag
DMAMUX_INT_FLG G_GENCH2_TO	DMAMUX request generator channel 2 trigger overrun interrupt flag
DMAMUX_INT_FLG G_GENCH3_TO	DMAMUX request generator channel 3 trigger overrun interrupt flag
DMAMUX_INT_FLG G_GENCH4_TO	DMAMUX request generator channel 4 trigger overrun interrupt flag
DMAMUX_INT_FLG G_GENCH5_TO	DMAMUX request generator channel 5 trigger overrun interrupt flag
DMAMUX_INT_FLG G_GENCH6_TO	DMAMUX request generator channel 6 trigger overrun interrupt flag

## dma\_deinit

The description of dma\_deinit is shown as below:

**Table 3-265. Function dma\_deinit**

Function name	dma_deinit
Function prototype	void dma_deinit(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	deinitialize DMA registers of a channel
Precondition	-
The called functions	-
Input parameter{in}	

<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. 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\_single\_data\_para\_struct\_init**

The description of dma\_single\_data\_para\_struct\_init is shown as below:

**Table 3-266. Function dma\_single\_data\_para\_struct\_init**

<b>Function name</b>	dma_single_data_para_struct_init
<b>Function prototype</b>	void dma_single_data_para_struct_init(dma_single_data_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize the DMA single data mode parameters struct 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 DMA channel, refer to <a href="#">Table 3-256. Structure dma_single_data_parameter_struct</a> .
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of DMA */
```

```
dma_single_data_parameter_struct init_struct;
```

```
dma_single_data_para_struct_init(&init_struct);
```

### **dma\_multi\_data\_para\_struct\_init**

The description of dma\_multi\_data\_para\_struct\_init is shown as below:

**Table 3-267. Function dma\_multi\_data\_para\_struct\_init**

<b>Function name</b>	dma_multi_data_para_struct_init
----------------------	---------------------------------

<b>Function prototype</b>	void dma_multi_data_para_struct_init(dma_multi_data_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize the DMA multi data mode parameters struct 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 DMA channel, refer to <a href="#">Table 3-255. Structure dma_multi_data_parameter_struct.</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of DMA */

dma_multi_data_parameter_struct init_struct;

dma_multi_data_para_struct_init(&init_struct);
```

### **dma\_single\_data\_mode\_init**

The description of dma\_single\_data\_mode\_init is shown as below:

**Table 3-268. Function dma\_single\_data\_mode\_init**

<b>Function name</b>	dma_single_data_mode_init
<b>Function prototype</b>	void dma_single_data_mode_init(uint32_t dma_periph, dma_channel_enum channelx, dma_single_data_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize DMA single data mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum.</a>
<b>Input parameter{in}</b>	
<b>init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-256. Structure dma_single_data_parameter_struct.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* initialize DMA1 channel7 */

dma_single_data_parameter_struct dma_init_struct;

dma_single_data_para_struct_init(&dma_init_struct);

dma_deinit(DMA1, DMA_CH7);

dma_init_struct.request = DMA_REQUEST_USART0_TX;

dma_init_struct.direction = DMA_MEMORY_TO_PERIPH;

dma_init_struct.memory0_addr = (uint32_t)txbuffer;

dma_init_struct.memory_inc = DMA_MEMORY_INCREASE_ENABLE;

dma_init_struct.periph_memory_width = DMA_PERIPH_WIDTH_8BIT;

dma_init_struct.number = ARRAYNUM(txbuffer);

dma_init_struct.periph_addr = USART0_TDATA_ADDRESS;

dma_init_struct.periph_inc = DMA_PERIPH_INCREASE_DISABLE;

dma_init_struct.priority = DMA_PRIORITY_ULTRA_HIGH;

dma_single_data_mode_init(DMA1, DMA_CH7, &dma_init_struct);

```

### **dma\_multi\_data\_mode\_init**

The description of dma\_multi\_data\_mode\_init is shown as below:

**Table 3-269. Function dma\_multi\_data\_mode\_init**

Function name	dma_multi_data_mode_init
Function prototype	void dma_multi_data_mode_init(uint32_t dma_periph, dma_channel_enum channelx, dma_multi_data_parameter_struct *init_struct);
Function descriptions	initialize DMA multi data mode
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Input parameter{in}	
init_struct	Structure for initialization, the structure members can refer to <a href="#">Table 3-255. Structure dma_multi_data_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize DMA1 channel 0 */

dma_multi_data_parameter_struct dma_init_parameter

dma_multi_data_para_struct_init(&dma_init_parameter);

dma_deinit(DMA1,DMA_CH0);

dma_init_parameter.periph_addr = (uint32_t)source;

dma_init_parameter.periph_width = DMA_PERIPH_WIDTH_16BIT;

dma_init_parameter.periph_inc = DMA_PERIPH_INCREASE_ENABLE;

dma_init_parameter.memory0_addr = (uint32_t)destination;

dma_init_parameter.memory_width = DMA_MEMORY_WIDTH_16BIT;

dma_init_parameter.memory_inc = DMA_MEMORY_INCREASE_ENABLE;

dma_init_parameter.memory_burst_width = DMA_MEMORY_BURST_4_BEAT;

dma_init_parameter.periph_burst_width = DMA_PERIPH_BURST_4_BEAT;

dma_init_parameter.critical_value = DMA_FIFO_2_WORD;

dma_init_parameter.circular_mode = DMA_CIRCULAR_MODE_DISABLE;

dma_init_parameter.direction = DMA_MEMORY_TO_MEMORY;

dma_init_parameter.number = BUFFER_SIZE;

dma_init_parameter.priority = DMA_PRIORITY_ULTRA_HIGH;

dma_multi_data_mode_init(DMA1,DMA_CH0,&dma_init_parameter);

```

## dma\_periph\_address\_config

The description of dma\_periph\_address\_config is shown as below:

**Table 3-270. 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>	configure 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .

Input parameter{in}	
address	peripheral base address, 0x00000000-0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 channel 0 peripheral base address */
dma_periph_address_config(DMA0, DMA_CH0, 0x08000000);
```

### **dma\_memory\_address\_config**

The description of dma\_memory\_address\_config is shown as below:

**Table 3-271. Function dma\_memory\_address\_config**

Function name	dma_memory_address_config
Function prototype	void dma_memory_address_config(uint32_t dma_periph, dma_channel_enum channelx, uint8_t memory_flag, uint32_t address);
Function descriptions	configure DMA memory base address
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Input parameter{in}	
memory_flag	DMA memory selection
DMA_MEMORY_0	DMA memory 0
DMA_MEMORY_1	DMA memory 1
Input parameter{in}	
address	memory base address, 0x00000000-0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable DMA0 channel 0 memory base address */
dma_memory_address_config(DMA0, DMA_CH0, DMA_MEMORY_0, 0x08000000);
```



## dma\_transfer\_number\_config

The description of dma\_transfer\_number\_config is shown as below:

**Table 3-272. Function dma\_transfer\_number\_config**

<b>Function name</b>	dma_transfer_number_config
<b>Function prototype</b>	void dma_transfer_number_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t number);
<b>Function descriptions</b>	configure 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>number</b>	the number of remaining data to be transferred by the DMA, 0x00000000-0x0000FFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the number of remaining data to be transferred by the DMA0 channel 0 */
```

```
dma_transfer_number_config(DMA0, DMA_CH0, 0xFF);
```

## dma\_transfer\_number\_get

The description of dma\_transfer\_number\_get is shown as below:

**Table 3-273. 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>	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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .

Output parameter{out}	
-	-
Return value	
uint32_t	the number of remaining data to be transferred by the DMA, 0x00000000-0x0000FFFF

Example:

```
/* get the number of remaining data to be transferred by the DMA0 channel 0 */
```

```
uint32_t data_num;
```

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

### **dma\_priority\_config**

The description of dma\_priority\_config is shown as below:

**Table 3-274. Function dma\_priority\_config**

Function name	dma_priority_config
Function prototype	void dma_priority_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t priority);
Function descriptions	configure priority level of DMA channel
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Input parameter{in}	
priority	priority level of this channel
DMA_PRIORITY_LOW	low priority
DMA_PRIORITY_MEDIUM	medium priority
DMA_PRIORITY_HIGH	high priority
DMA_PRIORITY_ULTRA_HIGH	ultra high priority
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure priority level of DMA0 channel 0 */
```

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

## dma\_memory\_burst\_beats\_config

The description of dma\_memory\_burst\_beats\_config is shown as below:

**Table 3-275. Function dma\_memory\_burst\_beats\_config**

<b>Function name</b>	dma_memory_burst_beats_config
<b>Function prototype</b>	void dma_memory_burst_beats_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t mbeat);
<b>Function descriptions</b>	configure transfer burst beats 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>mbeat</b>	memory transfer burst beats
<i>DMA_MEMORY_BURST_SINGLE</i>	memory transfer single burst
<i>DMA_MEMORY_BURST_4_BEAT</i>	memory transfer 4-beat burst
<i>DMA_MEMORY_BURST_8_BEAT</i>	memory transfer 8-beat burst
<i>DMA_MEMORY_BURST_16_BEAT</i>	memory transfer 16-beat burst
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure transfer burst beats of memory */
```

```
dma_memory_burst_beats_config(DMA0, DMA_CH0, DMA_MEMORY_BURST_4_BEAT);
```

## dma\_periph\_burst\_beats\_config

The description of dma\_periph\_burst\_beats\_config is shown as below:

**Table 3-276. Function dma\_periph\_burst\_beats\_config**

<b>Function name</b>	dma_periph_burst_beats_config
<b>Function prototype</b>	void dma_periph_burst_beats_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t pbeat);
<b>Function descriptions</b>	configure transfer burst beats 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum.</a>
<b>Input parameter{in}</b>	
<b>mbeat</b>	peripheral transfer burst beats
<i>DMA_PERIPH_BURST_SINGLE</i>	peripheral transfer single burst
<i>DMA_PERIPH_BURST_4_BEAT</i>	peripheral transfer 4-beat burst
<i>DMA_PERIPH_BURST_8_BEAT</i>	peripheral transfer 8-beat burst
<i>DMA_PERIPH_BURST_16_BEAT</i>	peripheral transfer 16-beat burst
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure transfer burst beats of peripheral */
```

```
dma_periph_burst_beats_config(DMA0, DMA_CH0, DMA_PERIPH_BURST_4_BEAT);
```

### **dma\_memory\_width\_config**

The description of dma\_memory\_width\_config is shown as below:

**Table 3-277. 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 msize);
<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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum.</a>
<b>Input parameter{in}</b>	
<b>msize</b>	transfer data size of memory

<i>DMA_MEMORY_WIDT H_8BIT</i>	transfer data size of memory is 8-bit
<i>DMA_MEMORY_WIDT H_16BIT</i>	transfer data size of memory is 16-bit
<i>DMA_MEMORY_WIDT H_32BIT</i>	transfer data size of memory is 32-bit
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure transfer data size of memory */
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 3-278. 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 psize);
<b>Function descriptions</b>	configure transfer data size 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>psize</b>	transfer data size of peripheral
<i>DMA_PERIPH_WIDTH _8BIT</i>	transfer data size of peripheral is 8-bit
<i>DMA_PERIPH_WIDTH _16BIT</i>	transfer data size of peripheral is 16-bit
<i>DMA_PERIPH_WIDTH _32BIT</i>	transfer data size of peripheral is 32-bit
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure transfer data size of peripheral */
```

```
dma_periph_width_config(DMA0, DMA_CH0, DMA_PERIPH_WIDTH_16BIT);
```

### **dma\_memory\_address\_generation\_config**

The description of dma\_memory\_address\_generation\_config is shown as below:

**Table 3-279. Function dma\_memory\_address\_generation\_config**

<b>Function name</b>	dma_memory_address_generation_config
<b>Function prototype</b>	void dma_memory_address_generation_config(uint32_t dma_periph, dma_channel_enum channelx, uint8_t generation_algorithm);
<b>Function descriptions</b>	configure memory address generation algorithm
<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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>generation_algorithm</b>	the address generation algorithm
<i>DMA_MEMORY_INCR EASE_ENABLE</i>	next address of memory is increasing address mode
<i>DMA_MEMORY_INCR EASE_DISABLE</i>	next address of memory is fixed address mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure memory address generation algorithm */
```

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

### **dma\_peripheral\_address\_generation\_config**

The description of dma\_peripheral\_address\_generation\_config is shown as below:

**Table 3-280. Function dma\_peripheral\_address\_generation\_config**

<b>Function name</b>	dma_peripheral_address_generation_config
<b>Function prototype</b>	void dma_peripheral_address_generation_config(uint32_t dma_periph, dma_channel_enum channelx, uint8_t generation_algorithm);

<b>Function descriptions</b>	configure peripheral address generation algorithm
<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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum.</a>
<b>Input parameter{in}</b>	
<b>generation_algorithm</b>	the address generation algorithm
<i>DMA_PERIPH_INCREASE_ENABLE</i>	next address of peripheral is increasing address mode
<i>DMA_PERIPH_INCREASE_DISABLE</i>	next address of peripheral is fixed address mode
<i>DMA_PERIPH_INCREASE_FIX</i>	increasing steps of peripheral address is fixed
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure peripheral address generation algorithm */
```

```
dma_peripheral_address_generation_config(DMA0, DMA_CH0, DMA_PERIPH_INCREASE_DISABLE);
```

### dma\_circulation\_enable

The description of dma\_circulation\_enable is shown as below:

**Table 3-281. 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum.</a>
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

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 3-282. Function dma\_circulation\_disable**

Function name	dma_circulation_disable
Function prototype	void dma_circulation_disable(uint32_t dma_periph, dma_channel_enum channelx);
Function descriptions	disable DMA circulation mode
Precondition	corresponding channel enable bit CHEN should be 0
The called functions	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable DMA0 channel 0 circulation mode */
```

```
dma_circulation_disable(DMA0, DMA_CH0);
```

### dma\_channel\_enable

The description of dma\_channel\_enable is shown as below:

**Table 3-283. Function dma\_channel\_enable**

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



<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. 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 3-284. 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMA channel 0 */
dma_channel_disable(DMA0, DMA_CH0);
```

### **dma\_transfer\_direction\_config**

The description of dma\_transfer\_direction\_config is shown as below:

**Table 3-285. 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, uint8_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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum.</a>
<b>Input parameter{in}</b>	
<b>direction</b>	specify the direction of data transfer
<i>DMA_PERIPH_TO_MEMORY</i>	read from peripheral and write to memory
<i>DMA_MEMORY_TO_PERIPH</i>	read from memory and write to peripheral
<i>DMA_MEMORY_TO_MEMORY</i>	read from memory and write to memory
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure DMA0 channel 0 memory to memory mode */
```

```
dma_memory_to_memory_enable(DMA0, DMA_CH0, DMA_MEMORY_TO_MEMORY);
```

### **dma\_switch\_buffer\_mode\_config**

The description of dma\_switch\_buffer\_mode\_config is shown as below:

**Table 3-286. Function dma\_switch\_buffer\_mode\_config**

<b>Function name</b>	dma_switch_buffer_mode_config
<b>Function prototype</b>	void dma_switch_buffer_mode_config(uint32_t dma_periph, dma_channel_enum channelx, uint32_t memory1_addr, uint32_t memory_select);
<b>Function descriptions</b>	configure DMA switch buffer 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum.</a>

Input parameter{in}	
<b>memory1_addr</b>	memory1 base address, 0x00000000-0xFFFFFFFF
Input parameter{in}	
<b>memory_select</b>	DMA memory selection
<i>DMA_MEMORY_0</i>	DMA memory 0
<i>DMA_MEMORY_1</i>	DMA memory 1
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
dma_switch_buffer_mode_config(DMA0, DMA_CH0, 0x20000000, DMA_MEMORY_0);
```

### dma\_using\_memory\_get

The description of dma\_using\_memory\_get is shown as below:

**Table 3-287. Function dma\_using\_memory\_get**

<b>Function name</b>	dma_using_memory_get
<b>Function prototype</b>	uint32_t dma_using_memory_get(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	get DMA using memory
<b>Precondition</b>	corresponding channel enable bit CHEN should be 0
<b>The called functions</b>	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Output parameter{out}	
-	-
Return value	
<b>uint32_t</b>	the using memory
<i>DMA_MEMORY_0</i>	DMA memory 0
<i>DMA_MEMORY_1</i>	DMA memory 1

Example:

```
/* get DMA using memory */
```

```
uint32_t memory_index;
```

```
memory_index = dma_using_memory_get(DMA0, DMA_CH0);
```

## dma\_switch\_buffer\_mode\_enable

The description of dma\_switch\_buffer\_mode\_enable is shown as below:

**Table 3-288. Function dma\_switch\_buffer\_mode\_enable**

<b>Function name</b>	dma_switch_buffer_mode_enable
<b>Function prototype</b>	void dma_switch_buffer_mode_enable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	enable DMA switch buffer 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DMA switch buffer mode */
dma_switch_buffer_mode_enable();
```

## dma\_switch\_buffer\_mode\_disable

The description of dma\_switch\_buffer\_mode\_disable is shown as below:

**Table 3-289. Function dma\_switch\_buffer\_mode\_disable**

<b>Function name</b>	dma_switch_buffer_mode_disable
<b>Function prototype</b>	void dma_switch_buffer_mode_disable(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	disable DMA switch buffer 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</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable DMA switch buffer mode */
dma_switch_buffer_mode_disable();
```

### **dma\_fifo\_status\_get**

The description of dma\_fifo\_status\_get is shown as below:

**Table 3-290. Function dma\_fifo\_status\_get**

<b>Function name</b>	dma_fifo_status_get
<b>Function prototype</b>	uint32_t dma_fifo_status_get(uint32_t dma_periph, dma_channel_enum channelx);
<b>Function descriptions</b>	get DMA FIFO status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the number of words stored in FIFO
<i>DMA_FIFO_CNT_NO_DATA</i>	no data
<i>DMA_FIFO_CNT_1_WORD</i>	1 word
<i>DMA_FIFO_CNT_2_WORD</i>	2 words
<i>DMA_FIFO_CNT_3_WORD</i>	3 words
<i>DMA_FIFO_CNT_EMPTY</i>	FIFO empty
<i>DMA_FIFO_CNT_FULL</i>	FIFO full

Example:

```
/* get DMA channel 0 transfer number */
uint32_t number = 0;
number = dma_transfer_number_get(DMA0, DMA_CH0);
```

## dma\_flag\_get

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

**Table 3-291. 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>	get DMA flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
<b>Input parameter{in}</b>	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>flag</b>	specify get which flag
<i>DMA_FLAG_FEE</i>	FIFO error and exception flag
<i>DMA_FLAG_SDE</i>	single data mode exception flag
<i>DMA_FLAG_TAE</i>	transfer access error flag
<i>DMA_FLAG_HTF</i>	half transfer finish flag
<i>DMA_FLAG_FTF</i>	full transfer finish flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	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 3-292. Function dma\_flag\_clear**

<b>Function name</b>	dma_flag_clear
<b>Function prototype</b>	void dma_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t flag);
<b>Function descriptions</b>	clear DMA flag
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Input parameter{in}	
<b>flag</b>	specify get which flag
<i>DMA_FLAG_FEE</i>	FIFO error and exception flag
<i>DMA_FLAG_SDE</i>	single data mode exception flag
<i>DMA_FLAG_TAE</i>	transfer access error flag
<i>DMA_FLAG_HTF</i>	half transfer finish flag
<i>DMA_FLAG_FTF</i>	full transger finish flag
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 3-293. Function dma\_interrupt\_enable**

<b>Function name</b>	dma_interrupt_enable
<b>Function prototype</b>	void dma_interrupt_enable(uint32_t dma_periph, dma_channel_enum channelx, uint32_t interrupt);
<b>Function descriptions</b>	enable DMA interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>dma_periph</b>	DMA peripheral
<i>DMAx</i>	(x=0,1)
Input parameter{in}	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Input parameter{in}	
<b>interrupt</b>	DMA interrupt
<i>DMA_INT_SDE</i>	single data mode exception interrupt
<i>DMA_INT_TAE</i>	half transfer finish interrupt of channel
<i>DMA_INT_HTF</i>	half transfer finish interrupt
<i>DMA_INT_FTF</i>	full transfer finish interrupt
<i>DMA_INT_FEE</i>	FIFO exception interrupt

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 3-294. 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 interrupt);
Function descriptions	disable DMA interrupt
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Input parameter{in}	
interrupt	DMA interrupt
DMA_INT_SDE	single data mode exception interrupt
DMA_INT_TAE	half transfer finish interrupt of channel
DMA_INT_HTF	half transfer finish interrupt
DMA_INT_FTF	full transfer finish interrupt
DMA_INT_FEE	FIFO exception interrupt
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 3-295. Function dma\_interrupt\_flag\_get

Function name	dma_interrupt_flag_get
Function prototype	FlagStatus dma_interrupt_flag_get(uint32_t dma_periph, dma_channel_enum channelx, uint32_t int_flag);
Function descriptions	get DMA interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	(x=0,1)
Input parameter{in}	
channelx	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Input parameter{in}	
int_flag	specify get which interrupt flag
DMA_INT_FLAG_FEE	FIFO error and exception interrupt flag
DMA_INT_FLAG_SDE	single data mode exception interrupt flag
DMA_INT_FLAG_TAE	transfer access error interrupt flag
DMA_INT_FLAG_HTF	half transfer finish interrupt flag
DMA_INT_FLAG_FTF	full transger finish interrupt flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMA0 channel 3 interrupt flag */
```

```
FlagStatus flag = RESET;
```

```
flag = dma_interrupt_flag_get(DMA0, DMA_CH3, DMA_INT_FTF);
```

### dma\_interrupt\_flag\_clear

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

Table 3-296. Function dma\_interrupt\_flag\_clear

Function name	dma_interrupt_flag_clear
Function prototype	void dma_interrupt_flag_clear(uint32_t dma_periph, dma_channel_enum channelx, uint32_t int_flag);
Function descriptions	clear DMA interrupt flag
Precondition	-
The called functions	-
Input parameter{in}	
dma_periph	DMA peripheral
DMAx	(x=0,1)

Input parameter{in}	
<b>channelx</b>	DMA channel, refer to <a href="#">Table 3-259. Enum dma_channel_enum</a> .
Input parameter{in}	
<b>int_flag</b>	specify get which interrupt flag
<i>DMA_INT_FLAG_FEE</i>	FIFO error and exception interrupt flag
<i>DMA_INT_FLAG_SDE</i>	single data mode exception interrupt flag
<i>DMA_INT_FLAG_TAE</i>	transfer access error interrupt flag
<i>DMA_INT_FLAG_HTF</i>	half transfer finish interrupt flag
<i>DMA_INT_FLAG_FTF</i>	full transfer finish interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear DMA0 channel 3 interrupt flag */
dma_interrupt_flag_clear(DMA0, DMA_CH3, DMA_INT_FTF);
```

### dmamux\_sync\_struct\_para\_init

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

**Table 3-297. 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>	-
Input parameter{in}	
-	-
Output parameter{out}	
<b>init_struct</b>	the initialization data needed to initialize DMAMUX request multiplexer channel synchronization mode, refer to <a href="#">Table 3-257. Structure dmamux_sync_parameter_struct</a> .
Return value	
-	-

Example:

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

## dmamux\_synchronization\_init

The description of dmamux\_synchronization\_init is shown as below:

**Table 3-298. 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>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. 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 3-257. 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 3-299. Function dmamux\_synchronization\_enable**

<b>Function name</b>	dmamux_synchronization_enable
<b>Function prototype</b>	void dmamux_synchronization_enable(dmamux_multiplexer_channel_enum channelx);
<b>Function descriptions</b>	enable synchronization mode

Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. 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 3-300. Function dmamux\_synchronization\_disable**

Function name	dmamux_synchronization_disable
Function prototype	void dmamux_synchronization_disable(dmamux_multiplexer_channel_enum channelx);
Function descriptions	disable synchronization mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. Enum dmamux_multiplexer_channel_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable synchronization mode */
```

```
dmamux_synchronization_disable(DMAMUX_MUXCH0);
```

### dmamux\_event\_generation\_enable

The description of dmamux\_event\_generation\_enable is shown as below:

**Table 3-301. Function dmamux\_event\_generation\_enable**

Function name	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>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. Enum dmamux_multiplexer_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable event generation */
```

```
dmamux_event_generation_enable(DMAMUX_MUXCH0);
```

### dmamux\_event\_generation\_disable

The description of dmamux\_event\_generation\_disable is shown as below:

**Table 3-302. Function dmamux\_event\_generation\_disable**

<b>Function name</b>	dmamux_event_generation_disable
<b>Function prototype</b>	void dmamux_event_generation_disable(dmamux_multiplexer_channel_enum channelx);
<b>Function descriptions</b>	disable event generation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. 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 3-303. 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 3-258. 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 3-304. Function dmamux\_request\_generator\_init**

<b>Function name</b>	dmamux_request_generator_init
<b>Function prototype</b>	void dmamux_request_generator_init(dmamux_generator_channel_enum channelx, dmamux_gen_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize DMAMUX request generator channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	DMAMUX generation channel, refer to <a href="#">Table 3-261. Enum dmamux_generator_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>init_struct</b>	the initialization data needed to initialize DMAMUX request generator channel, refer to <a href="#">Table 3-258. Structure dmamux_gen_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-

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 3-305. Function dmamux\_request\_generator\_channel\_enable**

Function name	dmamux_request_generator_channel_enable
Function prototype	void dmamux_request_generator_channel_enable(dmamux_generator_channel_enum channelx);
Function descriptions	enable DMAMUX request generator channel
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX generation channel, refer to <a href="#">Table 3-261. Enum dmamux_generator_channel_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* enable DMAMUX request generator channel 0 */

dmamux_request_generator_channel_enable(DMAMUX_GENCH0);

```

### dmamux\_request\_generator\_channel\_disable

The description of dmamux\_request\_generator\_channel\_disable is shown as below:

**Table 3-306. Function dmamux\_request\_generator\_channel\_disable**

Function name	dmamux_request_generator_channel_disable
---------------	--

<b>Function prototype</b>	void dmamux_request_generator_channel_disable(dmamux_generator_channel_enum channelx);
<b>Function descriptions</b>	disable DMAMUX request generator channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	DMAMUX generation channel, refer to <a href="#">Table 3-261. Enum dmamux_generator_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMAMUX request generator channel 0 */
```

```
dmamux_request_generator_channel_disable(DMAMUX_GENCH0);
```

### dmamux\_synchronization\_polarity\_config

The description of dmamux\_synchronization\_polarity\_config is shown as below:

**Table 3-307. Function dmamux\_synchronization\_polarity\_config**

<b>Function name</b>	dmamux_synchronization_polarity_config
<b>Function prototype</b>	void dmamux_synchronization_polarity_config(dmamux_multiplexer_channel_enum channelx, uint32_t polarity);
<b>Function descriptions</b>	configure synchronization input polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. Enum dmamux_multiplexer_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>polarity</b>	synchronization input polarity
DMAMUX_SYNC_NO_EVENT	no event detection
DMAMUX_SYNC_RISING	rising edge
DMAMUX_SYNC_FALLING	falling edge
DMAMUX_SYNC_RISING_FALLING	rising and falling edges
<b>Output parameter{out}</b>	



-	-
Return value	
-	-

Example:

```
/* configure synchronization input polarity */
```

```
dmamux_synchronization_polarity_config(DMAMUX_MUXCH0, DMAMUX_SYNC_RISING);
```

### dmamux\_request\_forward\_number\_config

The description of dmamux\_request\_forward\_number\_config is shown as below:

**Table 3-308. 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>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. Enum dmamux_multiplexer_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>number</b>	DMA requests number to forward (1 - 32)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-309. Function dmamux\_sync\_id\_config**

<b>Function name</b>	dmamux_sync_id_config
<b>Function prototype</b>	void dmamux_sync_id_config(dmamux_multiplexer_channel_enum channelx, uint32_t id);
<b>Function descriptions</b>	configure synchronization input identification
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
channelx	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. 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
DMAMUX_SYNC_EXTI10	synchronization input is EXTI10
DMAMUX_SYNC_EXTI11	synchronization input is EXTI11
DMAMUX_SYNC_EXTI12	synchronization input is EXTI12
DMAMUX_SYNC_EXTI13	synchronization input is EXTI13
DMAMUX_SYNC_EXTI14	synchronization input is EXTI14
DMAMUX_SYNC_EXTI15	synchronization input is EXTI15
DMAMUX_SYNC_EVTX_OUT0	synchronization input is Evt_out0
DMAMUX_SYNC_EVTX_OUT1	synchronization input is Evt_out1
DMAMUX_SYNC_EVTX_OUT2	synchronization input is Evt_out2

<code>X_OUT2</code>	
<code>DMAMUX_SYNC_EVT</code> <code>X_OUT3</code>	synchronization input is Evt_out3
<code>DMAMUX_SYNC_EVT</code> <code>X_OUT4</code>	synchronization input is Evt_out4
<code>DMAMUX_SYNC_EVT</code> <code>X_OUT5</code>	synchronization input is Evt_out5
<code>DMAMUX_SYNC_EVT</code> <code>X_OUT6</code>	synchronization input is Evt_out6
<code>DMAMUX_SYNC_RTC</code> <code>_WAKEUP</code>	synchronization input is RTC wakeup
<code>DMAMUX_SYNC_CMP</code> <code>0_OUTPUT</code>	synchronization input is CMP0 output
<code>DMAMUX_SYNC_I2C0</code> <code>_WAKEUP</code>	synchronization input is I2C0 wakeup
<code>DMAMUX_SYNC_I2C1</code> <code>_WAKEUP</code>	synchronization input is I2C1 wakeup
<code>DMAMUX_SYNC_I2C2</code> <code>_WAKEUP</code>	synchronization input is I2C2 wakeup
<code>DMAMUX_SYNC_I2C3</code> <code>_WAKEUP</code>	synchronization input is I2C3 wakeup
<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 3-310. 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>	DMAMUX request multiplexer channel, refer to <a href="#">Table 3-260. Enum dmamux_multiplexer_channel_enum</a> .

Input parameter{in}	
<b>id</b>	input DMA request identification
<i>DMA_REQUEST_M2M</i>	memory to memory transfer
<i>DMA_REQUEST_GENERATOR0</i>	DMAMUX request generator 0
<i>DMA_REQUEST_GENERATOR1</i>	DMAMUX request generator 1
<i>DMA_REQUEST_GENERATOR2</i>	DMAMUX request generator 2
<i>DMA_REQUEST_GENERATOR3</i>	DMAMUX request generator 3
<i>DMA_REQUEST_GENERATOR4</i>	DMAMUX request generator 4
<i>DMA_REQUEST_GENERATOR5</i>	DMAMUX request generator 5
<i>DMA_REQUEST_GENERATOR6</i>	DMAMUX request generator 6
<i>DMA_REQUEST_GENERATOR7</i>	DMAMUX request generator 7
<i>DMA_REQUEST_ADC0</i>	DMAMUX ADC0 request
<i>DMA_REQUEST_ADC1</i>	DMAMUX ADC1 request
<i>DMA_REQUEST_TIMER0_CH0</i>	DMAMUX TIMER0 CH0 request
<i>DMA_REQUEST_TIMER0_CH1</i>	DMAMUX TIMER0 CH1 request
<i>DMA_REQUEST_TIMER0_CH2</i>	DMAMUX TIMER0 CH2 request
<i>DMA_REQUEST_TIMER0_CH3</i>	DMAMUX TIMER0 CH3 request
<i>DMA_REQUEST_TIMER0_MCH0</i>	DMAMUX TIMER0 MCH0 request
<i>DMA_REQUEST_TIMER0_MCH1</i>	DMAMUX TIMER0 MCH1 request
<i>DMA_REQUEST_TIMER0_MCH2</i>	DMAMUX TIMER0 MCH2 request
<i>DMA_REQUEST_TIMER0_MCH3</i>	DMAMUX TIMER0 MCH3 request
<i>DMA_REQUEST_TIMER0_UP</i>	DMAMUX TIMER0 UP request
<i>DMA_REQUEST_TIMER0_TRG</i>	DMAMUX TIMER0 TRG request

<i>DMA_REQUEST_TIME</i> <i>R0_CMT</i>	DMAMUX TIMER0 CMT request
<i>DMA_REQUEST_TIME</i> <i>R1_CH0</i>	DMAMUX TIMER1 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH1</i>	DMAMUX TIMER1 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH2</i>	DMAMUX TIMER1 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R1_CH3</i>	DMAMUX TIMER1 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R1_UP</i>	DMAMUX TIMER1 UP request
<i>DMA_REQUEST_TIME</i> <i>R1_TRG</i>	DMAMUX TIMER1 TRG 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_TIME</i> <i>R2_UP</i>	DMAMUX TIMER2 UP request
<i>DMA_REQUEST_TIME</i> <i>R2_TRG</i>	DMAMUX TIMER2 TRG request
<i>DMA_REQUEST_TIME</i> <i>R3_CH0</i>	DMAMUX TIMER3 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R3_CH1</i>	DMAMUX TIMER3 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R3_CH2</i>	DMAMUX TIMER3 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R3_CH3</i>	DMAMUX TIMER3 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R3_CH3</i>	DMAMUX TIMER3 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R3_TRG</i>	DMAMUX TIMER3 TRG request
<i>DMA_REQUEST_TIME</i> <i>R3_UP</i>	DMAMUX TIMER3 UP request
<i>DMA_REQUEST_I2C0</i> <i>_RX</i>	DMAMUX I2C0 RX request
<i>DMA_REQUEST_I2C0</i>	DMAMUX I2C0 TX request

<i>_TX</i>	
<i>DMA_REQUEST_I2C1_RX</i>	DMAMUX I2C1 RX request
<i>DMA_REQUEST_I2C1_TX</i>	DMAMUX I2C1 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_SPI1_RX</i>	DMAMUX SPI1 RX request
<i>DMA_REQUEST_SPI1_TX</i>	DMAMUX SPI1 TX request
<i>DMA_REQUEST_USA_RT0_RX</i>	DMAMUX USART0 RX request
<i>DMA_REQUEST_USA_RT0_TX</i>	DMAMUX USART0 TX request
<i>DMA_REQUEST_USA_RT1_RX</i>	DMAMUX USART1 RX request
<i>DMA_REQUEST_USA_RT1_TX</i>	DMAMUX USART1 TX request
<i>DMA_REQUEST_USA_RT2_RX</i>	DMAMUX USART2 RX request
<i>DMA_REQUEST_USA_RT2_TX</i>	DMAMUX USART2 TX 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_R7_MCH0</i>	DMAMUX TIMER7 MCH0 request
<i>DMA_REQUEST_TIME_R7_MCH1</i>	DMAMUX TIMER7 MCH1 request
<i>DMA_REQUEST_TIME_R7_MCH2</i>	DMAMUX TIMER7 MCH2 request
<i>DMA_REQUEST_TIME_R7_MCH3</i>	DMAMUX TIMER7 MCH3 request
<i>DMA_REQUEST_TIME_R7_UP</i>	DMAMUX TIMER7 UP request

<i>DMA_REQUEST_TIME</i> <i>R7_TRG</i>	DMAMUX TIMER7 TRG request
<i>DMA_REQUEST_TIME</i> <i>R7_CMT</i>	DMAMUX TIMER7 CMT request
<i>DMA_REQUEST_TIME</i> <i>R4_CH0</i>	DMAMUX TIMER4 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R4_CH1</i>	DMAMUX TIMER4 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R4_CH2</i>	DMAMUX TIMER4 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R4_CH3</i>	DMAMUX TIMER4 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R4_UP</i>	DMAMUX TIMER4 UP request
<i>DMA_REQUEST_TIME</i> <i>R4_TRG</i>	DMAMUX TIMER4 TRG request
<i>DMA_REQUEST_SPI2</i> <i>_RX</i>	DMAMUX SPI2 RX request
<i>DMA_REQUEST_SPI2</i> <i>_TX</i>	DMAMUX SPI2 TX request
<i>DMA_REQUEST_UAR</i> <i>T3_RX</i>	DMAMUX UART3 RX request
<i>DMA_REQUEST_UAR</i> <i>T3_TX</i>	DMAMUX UART3 TX request
<i>DMA_REQUEST_UAR</i> <i>T4_RX</i>	DMAMUX UART4 RX request
<i>DMA_REQUEST_UAR</i> <i>T4_TX</i>	DMAMUX UART4 TX request
<i>DMA_REQUEST_DAC</i> <i>_CH0</i>	DMAMUX DAC CH0 request
<i>DMA_REQUEST_DAC</i> <i>_CH1</i>	DMAMUX DAC CH1 request
<i>DMA_REQUEST_TIME</i> <i>R5_UP</i>	DMAMUX TIMER5 UP request
<i>DMA_REQUEST_TIME</i> <i>R6_UP</i>	DMAMUX TIMER6 UP request
<i>DMA_REQUEST_USA</i> <i>RT5_RX</i>	DMAMUX USART5 RX request
<i>DMA_REQUEST_USA</i> <i>RT5_TX</i>	DMAMUX USART5 TX request
<i>DMA_REQUEST_I2C2</i> <i>_RX</i>	DMAMUX I2C2 RX request
<i>DMA_REQUEST_I2C2</i>	DMAMUX I2C2 TX request

<i>_TX</i>	
<i>DMA_REQUEST_UAR T6_RX</i>	DMAMUX UART6 RX request
<i>DMA_REQUEST_UAR T6_TX</i>	DMAMUX UART6 TX request
<i>DMA_REQUEST_UAR T7_RX</i>	DMAMUX UART7 RX request
<i>DMA_REQUEST_UAR T7_TX</i>	DMAMUX UART7 TX request
<i>DMA_REQUEST_SPI3 _RX</i>	DMAMUX SPI3 RX request
<i>DMA_REQUEST_SPI3 _TX</i>	DMAMUX SPI3 TX request
<i>DMA_REQUEST_SPI4 _RX</i>	DMAMUX SPI4 RX request
<i>DMA_REQUEST_SPI4 _TX</i>	DMAMUX SPI4 TX request
<i>DMA_REQUEST_HPD F_FLT0</i>	DMAMUX HPDF FLT0 request
<i>DMA_REQUEST_HPD F_FLT1</i>	DMAMUX HPDF FLT1 request
<i>DMA_REQUEST_HPD F_FLT2</i>	DMAMUX HPDF FLT2 request
<i>DMA_REQUEST_HPD F_FLT3</i>	DMAMUX HPDF FLT3 request
<i>DMA_REQUEST_TIME R14_CH0</i>	DMAMUX TIMER14 CH0 request
<i>DMA_REQUEST_TIME R14_CH1</i>	DMAMUX TIMER14 CH1 request
<i>DMA_REQUEST_TIME R14_MCH0</i>	DMAMUX TIMER14 MCH0 request
<i>DMA_REQUEST_TIME R14_UP</i>	DMAMUX TIMER14 UP request
<i>DMA_REQUEST_TIME R14_TRG</i>	DMAMUX TIMER14 TRG request
<i>DMA_REQUEST_TIME R14_CMT</i>	DMAMUX TIMER14 CMT request
<i>DMA_REQUEST_TIME R15_CH0</i>	DMAMUX TIMER15 CH0 request
<i>DMA_REQUEST_TIME R15_MCH0</i>	DMAMUX TIMER15 MCH0 request
<i>DMA_REQUEST_TIME R15_UP</i>	DMAMUX TIMER15 UP request



<i>DMA_REQUEST_TIME</i> <i>R16_CH0</i>	DMAMUX TIMER16 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R16_MCH0</i>	DMAMUX TIMER16 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R16_UP</i>	DMAMUX TIMER16 TRG request
<i>DMA_REQUEST_ADC</i> <i>2</i>	DMAMUX ADC2 request
<i>DMA_REQUEST_FAC</i> <i>_READ</i>	DMAMUX FAC READ request
<i>DMA_REQUEST_FAC</i> <i>_WRITE</i>	DMAMUX FAC WRITE request
<i>DMA_REQUEST_TMU</i> <i>_INPUT</i>	DMAMUX TMU INPUT request
<i>DMA_REQUEST_TMU</i> <i>_OUTPUT</i>	DMAMUX TMU OUTPUT request
<i>DMA_REQUEST_TIME</i> <i>R22_CH0</i>	DMAMUX TIMER22 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R22_CH1</i>	DMAMUX TIMER22 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R22_CH2</i>	DMAMUX TIMER22 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R22_CH3</i>	DMAMUX TIMER22 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R22_UP</i>	DMAMUX TIMER22 UP request
<i>DMA_REQUEST_TIME</i> <i>R22_TRG</i>	DMAMUX TIMER22 TRG request
<i>DMA_REQUEST_TIME</i> <i>R23_CH0</i>	DMAMUX TIMER23 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R23_CH1</i>	DMAMUX TIMER23 CH1 request
<i>DMA_REQUEST_TIME</i> <i>R23_CH2</i>	DMAMUX TIMER23 CH2 request
<i>DMA_REQUEST_TIME</i> <i>R23_CH3</i>	DMAMUX TIMER23 CH3 request
<i>DMA_REQUEST_TIME</i> <i>R23_UP</i>	DMAMUX TIMER23 UP request
<i>DMA_REQUEST_TIME</i> <i>R23_TRG</i>	DMAMUX TIMER23 TRG request
<i>DMA_REQUEST_TIME</i> <i>R40_CH0</i>	DMAMUX TIMER40 CH0 request
<i>DMA_REQUEST_TIME</i>	DMAMUX TIMER40 MCH0 request

<i>R40_MCH0</i>	
<i>DMA_REQUEST_TIME</i> <i>R40_CMT</i>	DMAMUX TIMER40 CMT request
<i>DMA_REQUEST_TIME</i> <i>R40_UP</i>	DMAMUX TIMER40 UP request
<i>DMA_REQUEST_TIME</i> <i>R41_CH0</i>	DMAMUX TIMER41 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R41_MCH0</i>	DMAMUX TIMER41 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R41_CMT</i>	DMAMUX TIMER41 CMT request
<i>DMA_REQUEST_TIME</i> <i>R41_UP</i>	DMAMUX TIMER41 UP request
<i>DMA_REQUEST_TIME</i> <i>R42_CH0</i>	DMAMUX TIMER42 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R42_MCH0</i>	DMAMUX TIMER42 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R42_CMT</i>	DMAMUX TIMER42 CMT request
<i>DMA_REQUEST_TIME</i> <i>R42_UP</i>	DMAMUX TIMER42 UP request
<i>DMA_REQUEST_TIME</i> <i>R43_CH0</i>	DMAMUX TIMER43 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R43_MCH0</i>	DMAMUX TIMER43 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R43_CMT</i>	DMAMUX TIMER43 CMT request
<i>DMA_REQUEST_TIME</i> <i>R43_UP</i>	DMAMUX TIMER43 UP request
<i>DMA_REQUEST_TIME</i> <i>R44_CH0</i>	DMAMUX TIMER44 CH0 request
<i>DMA_REQUEST_TIME</i> <i>R44_MCH0</i>	DMAMUX TIMER44 MCH0 request
<i>DMA_REQUEST_TIME</i> <i>R44_CMT</i>	DMAMUX TIMER44 CMT request
<i>DMA_REQUEST_TIME</i> <i>R44_UP</i>	DMAMUX TIMER44 UP request
<i>DMA_REQUEST_TIME</i> <i>R50_UP</i>	DMAMUX TIMER50 UP request
<i>DMA_REQUEST_TIME</i> <i>R51_UP</i>	DMAMUX TIMER51 UP request
<i>DMA_REQUEST_SPI5</i> <i>_RX</i>	DMAMUX SPI5 RX request

<i>DMA_REQUEST_SPI5_TX</i>	DMAMUX SPI5 TX request
<i>DMA_REQUEST_I2C3_RX</i>	DMAMUX I2C3 RX request
<i>DMA_REQUEST_I2C3_TX</i>	DMAMUX I2C3 TX request
<i>DMA_REQUEST_CAN0</i>	DMAMUX CAN0 request
<i>DMA_REQUEST_CAN1</i>	DMAMUX CAN1 request
<i>DMA_REQUEST_CAN2</i>	DMAMUX CAN2 request
<i>DMA_REQUEST_TIMER40_CH1</i>	DMAMUX TIMER40 CH1 request
<i>DMA_REQUEST_TIMER40_TRG</i>	DMAMUX TIMER40 TRG request
<i>DMA_REQUEST_TIMER41_CH1</i>	DMAMUX TIMER41 CH1 request
<i>DMA_REQUEST_TIMER41_TRG</i>	DMAMUX TIMER41 TRG request
<i>DMA_REQUEST_TIMER42_CH1</i>	DMAMUX TIMER42 CH1 request
<i>DMA_REQUEST_TIMER42_TRG</i>	DMAMUX TIMER42 TRG request
<i>DMA_REQUEST_TIMER43_CH1</i>	DMAMUX TIMER43 CH1 request
<i>DMA_REQUEST_TIMER43_TRG</i>	DMAMUX TIMER43 TRG request
<i>DMA_REQUEST_TIMER44_CH1</i>	DMAMUX TIMER44 CH1 request
<i>DMA_REQUEST_TIMER44_TRG</i>	DMAMUX TIMER44 TRG 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 3-311. Function dmamux\_trigger\_polarity\_config

Function name	dmamux_trigger_polarity_config
Function prototype	void dmamux_trigger_polarity_config(dmamux_generator_channel_enum channelx, uint32_t polarity);
Function descriptions	configure trigger input polarity
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX generation channel, refer to <a href="#">Table 3-261. Enum dmamux_generator_channel_enum</a> .
Input parameter{in}	
polarity	trigger input polarity
DMAMUX_GEN_NO_EVENT	no event detection
DMAMUX_GEN_RISING	rising edge
DMAMUX_GEN_FALLING	falling edge
DMAMUX_GEN_RISING_FALLING	rising and falling edges
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure trigger input polarity */
```

```
dmamux_trigger_polarity_config(DMAMUX_GENCH0, DMAMUX_GEN_RISING);
```

### dmamux\_request\_generate\_number\_config

The description of dmamux\_request\_generate\_number\_config is shown as below:

Table 3-312. Function dmamux\_request\_generate\_number\_config

Function name	dmamux_request_generate_number_config
Function prototype	void dmamux_request_generate_number_config(dmamux_generator_channel_enum channelx, uint32_t number);
Function descriptions	configure number of DMA requests to be generated
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX generation channel, refer to <a href="#">Table 3-261. 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 3-313. Function dmamux\_trigger\_id\_config**

Function name	dmamux_trigger_id_config
Function prototype	void dmamux_trigger_id_config(dmamux_generator_channel_enum channelx, uint32_t id);
Function descriptions	configure trigger input identification
Precondition	-
The called functions	-
Input parameter{in}	
channelx	DMAMUX generation channel, refer to <a href="#">Table 3-261. Enum dmamux_generator_channel_enum</a> .
Input parameter{in}	
id	trigger input identification
DMAMUX_TRIGGER_EXTI0	trigger input is EXTI0
DMAMUX_TRIGGER_EXTI1	trigger input is EXTI1
DMAMUX_TRIGGER_EXTI2	trigger input is EXTI2
DMAMUX_TRIGGER_EXTI3	trigger input is EXTI3
DMAMUX_TRIGGER_EXTI4	trigger input is EXTI4
DMAMUX_TRIGGER_EXTI5	trigger input is EXTI5
DMAMUX_TRIGGER_EXTI6	trigger input is EXTI6
DMAMUX_TRIGGER_EXTI7	trigger input is EXTI7
DMAMUX_TRIGGER_EXTI8	trigger input is EXTI8

<i>EXTI8</i>	
<i>DMAMUX_TRIGGER_</i> <i>EXTI9</i>	trigger input is EXTI9
<i>DMAMUX_TRIGGER_</i> <i>EXTI10</i>	trigger input is EXTI10
<i>DMAMUX_TRIGGER_</i> <i>EXTI11</i>	trigger input is EXTI11
<i>DMAMUX_TRIGGER_</i> <i>EXTI12</i>	trigger input is EXTI12
<i>DMAMUX_TRIGGER_</i> <i>EXTI13</i>	trigger input is EXTI13
<i>DMAMUX_TRIGGER_</i> <i>EXTI14</i>	trigger input is EXTI14
<i>DMAMUX_TRIGGER_</i> <i>EXTI15</i>	trigger input is EXTI15
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT0</i>	trigger input is Evt_out0
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT1</i>	trigger input is Evt_out1
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT2</i>	trigger input is Evt_out2
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT3</i>	trigger input is Evt_out3
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT4</i>	trigger input is Evt_out4
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT5</i>	trigger input is Evt_out5
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT6</i>	trigger input is Evt_out6
<i>DMAMUX_TRIGGER_</i> <i>EVT_OUT7</i>	trigger input is Evt_out7
<i>DMAMUX_TRIGGER_</i> <i>RTC_WAKEUP</i>	trigger input is wakeup
<i>DMAMUX_TRIGGER_</i> <i>CMP0_OUTPUT</i>	trigger input is CMP0 output
<i>DMAMUX_TRIGGER_</i> <i>CMP1_OUTPUT</i>	trigger input is CMP1 output
<i>DMAMUX_TRIGGER_I</i> <i>2C0_WAKEUP</i>	trigger input is I2C0 wakeup
<i>DMAMUX_TRIGGER_I</i> <i>2C1_WAKEUP</i>	trigger input is I2C1 wakeup
<i>DMAMUX_TRIGGER_I</i> <i>2C2_WAKEUP</i>	trigger input is I2C2 wakeup

DMAMUX_TRIGGER_I 2C3_WAKEUP	trigger input is I2C3 wakeup
DMAMUX_TRIGGER_I 2C0_INT_EVENT	trigger input is I2C0 interrupt event
DMAMUX_TRIGGER_I 2C1_INT_EVENT	trigger input is I2C1 interrupt event
DMAMUX_TRIGGER_I 2C2_INT_EVENT	trigger input is I2C2 interrupt event
DMAMUX_TRIGGER_I 2C3_INT_EVENT	trigger input is I2C3 interrupt event
DMAMUX_TRIGGER_ ADC2_INT	ADC2 interrupt
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 3-314. 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 3-263</a> .
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get DMAMUX flag */
```

```
FlagStatus flag = RESET;
```

```
flag = dmamux_flag_get(DMAMUX_FLAG_GENCH0_TO);
```

## dmamux\_flag\_clear

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

**Table 3-315. 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 3-263</a> .
<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 3-316. 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, refer to <a href="#">Table 3-262. Enum dmamux_interrupt_enum</a> .
<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 3-317. 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>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	specify which interrupt to disable, refer to <a href="#">Table 3-262. Enum dmamux_interrupt_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-318. Function dmamux\_interrupt\_flag\_get**

<b>Function name</b>	dmamux_interrupt_flag_get
<b>Function prototype</b>	FlagStatus dmamux_interrupt_flag_get(dmamux_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	get DMAMUX interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	flag type, refer to <a href="#">Table 3-264. Enum dmamux_interrupt_flag_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get DMAMUX interrupt flag */

FlagStatus flag = RESET;

flag = dmamux_interrupt_flag_get(DMAMUX_INT_FLAG_MUXCH0_SO);
```

## dmamux\_interrupt\_flag\_clear

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

**Table 3-319. 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>	-
<b>Input parameter{in}</b>	
<b>int_flag</b>	flag type, refer to <a href="#">Table 3-264. Enum dmamux_interrupt_flag_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear DMAMUX interrupt flag */
```

```
dmamux_interrupt_flag_clear(DMAMUX_INT_FLAG_MUXCH0_SO);
```

## 3.11. EDOUT

EDOUT is the encoder divided-output controller in the MCU. It is used to output location information obtained from the encoder in the form of A-phase, B-phase, and Z-phase pulses. The EDOUT registers are listed in chapter [3.11.1](#), the EDOUT firmware functions are introduced in chapter [3.11.2](#).

### 3.11.1. Descriptions of Peripheral registers

EDOUT registers are listed in the table shown as below:

**Table 3-320. EDOUT Registers**

Registers	Descriptions
EDOUT_CTL	EDOUT control register
EDOUT_ENABLE	EDOUT enable register
EDOUT_LOC	EDOUT location register
EDOUT_OCNT	EDOUT output counter register
EDOUT_LCNT	EDOUT location counter register
EDOUT_ZCR	EDOUT Z-phase configure register

### 3.11.2. Descriptions of Peripheral functions

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

**Table 3-321. EDOUT firmware function**

Function name	Function description
edout_deinit	deinitialize EDOUT
edout_init	initialize EDOUT
edout_enable	enable EDOUT
edout_disable	disable EDOUT
edout_polarity_config	set B-phase active polarity
edout_max_location_value_config	set the maximum location value for one rotation
edout_output_counter_update	update the output counter, used to set the phase difference and the number of edges for the next update period
edout_current_location_config	set the current location value
edout_current_location_get	get the current location value
edout_z_output_mode_config	configure Z-phase output mode
edout_z_output_start_loc_and_width_config	configure Z-phase output start location and width

#### edout\_deinit

The description of edout\_deinit is shown as below:

**Table 3-322. Function edout\_deinit**

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

Example:

```
/* deinitialize EDOUT */
edout_deinit();
```

#### edout\_init

The description of edout\_init is shown as below:

Table 3-323. Function edout\_init

Function name	edout_init
Function prototype	void edout_init(uint32_t pol, uint32_t max_loc, uint32_t cur_loc);
Function descriptions	initialize EDOUT
Precondition	-
The called functions	-
Input parameter{in}	
pol	the active polarity of the B-phase output signal selection
Input parameter{in}	
max_loc	(max_loc+1) must be a multiple of four between 16~65536 (e.g. 0x000F: The maximum location value is 16 (16=4*4))
Input parameter{in}	
cur_loc	current location value, 0~locmax (locmax is the LOCMAX bit fields value of EDOUT_LOC register)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
#define EDOUT_MAX_LOC          99
#define EDOUT_CUR_LOC          50
#define EDOUT_B_PHASE_POL      EDOUT_POL_POSITIVE

/* initialize EDOUT */
edout_init(EDOUT_B_PHASE_POL, EDOUT_MAX_LOC, EDOUT_CUR_LOC);
```

### edout\_enable

The description of edout\_enable is shown as below:

Table 3-324. Function edout\_enable

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

Example:

```
/* enable EDOUT */
edout_enable();
```

### edout\_disable

The description of edout\_disable is shown as below:

**Table 3-325. Function edout\_disable**

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

Example:

```
/* disable EDOUT */
edout_disable();
```

### edout\_polarity\_config

The description of edout\_polarity\_config is shown as below:

**Table 3-326. Function edout\_polarity\_config**

<b>Function name</b>	edout_polarity_config
<b>Function prototype</b>	void edout_polarity_config(uint32_t pol);
<b>Function descriptions</b>	set B-phase active polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pol</b>	the active polarity of the B-phase output signal selection
<i>EDOUT_POL_POSITIVE</i>	active polarity is positive
<i>EDOUT_POL_NEGATIVE</i>	active polarity is negative
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure B-phase active polarity */
edout_polarity_config(EDOUT_POL_POSITIVE);
```

### edout\_max\_location\_value\_config

The description of edout\_max\_location\_value\_config is shown as below:

**Table 3-327. Function edout\_max\_location\_value\_config**

<b>Function name</b>	edout_max_location_value_config
<b>Function prototype</b>	void edout_max_location_value_config(uint32_t max_loc);
<b>Function descriptions</b>	set the maximum location value for one rotation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>max_loc</b>	(max_loc+1) must be a multiple of four between 16~65536, e.g. 0x000F: The maximum location value is 16.
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the maximum location value of EDOUT */
edout_max_location_value_config(99);
```

### edout\_output\_counter\_update

The description of edout\_output\_counter\_update is shown as below:

**Table 3-328. Function edout\_output\_counter\_update**

<b>Function name</b>	edout_output_counter_update
<b>Function prototype</b>	void edout_output_counter_update(int16_t num_edges, uint16_t phase_diff);
<b>Function descriptions</b>	update the output counter, used to set the phase difference and the number of edges for the next update period
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>num_edges</b>	edge count, value range is -32768~32767, positive means clockwise rotation, negative means counter-clockwise rotation
<b>Input parameter{in}</b>	
<b>phase_diff</b>	phase difference, value range is 2~65535, in units of PCLK
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* set the edge count and the phase difference value between the phase-A and phase-B */
edout_output_counter_update(5, 0x7530);
```

### edout\_current\_location\_config

The description of edout\_current\_location\_config is shown as below:

**Table 3-329. Function edout\_current\_location\_config**

<b>Function name</b>	edout_current_location_config
<b>Function prototype</b>	void edout_current_location_config(uint32_t cur_loc);
<b>Function descriptions</b>	set the current location value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cur_loc</b>	current location value, 0~locmax (locmax is the LOCMAX bit fields value of EDOUT_LOC register)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the current location */
edout_current_location_config(90);
```

### edout\_current\_location\_get

The description of edout\_current\_location\_get is shown as below:

**Table 3-330. Function edout\_current\_location\_get**

<b>Function name</b>	edout_current_location_get
<b>Function prototype</b>	uint16_t edout_current_location_get(void)
<b>Function descriptions</b>	get the current location value
<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>	current location value, 0~locmax (locmax is the LOCMAX bit fields value of EDOUT_LOC register)

Example:

```
uint16_t cur_loc;

/* get the current location */
cur_loc = edout_current_location_get();
```

### edout\_z\_output\_mode\_config

The description of edout\_z\_output\_mode\_config is shown as below:

**Table 3-331. Function edout\_z\_output\_mode\_config**

<b>Function name</b>	edout_z_output_mode_config
<b>Function prototype</b>	void edout_z_output_mode_config(uint32_t mode);
<b>Function descriptions</b>	configure Z-phase output mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mode</b>	Z-phase output mode
EDOUT_Z_OUTPUT_MODE0	output according to the current location
EDOUT_Z_OUTPUT_MODE1	output according to the number of edges
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure Z-phase output mode */
edout_z_output_mode_config(EDOUT_Z_OUTPUT_MODE0);
```

### edout\_z\_output\_start\_loc\_and\_width\_config

The description of edout\_z\_output\_start\_loc\_and\_width\_config is shown as below:

**Table 3-332. Function edout\_z\_output\_start\_loc\_and\_width\_config**

<b>Function name</b>	edout_z_output_start_loc_and_width_config
<b>Function prototype</b>	void edout_z_output_start_loc_and_width_config(uint32_t start_loc, uint32_t width);
<b>Function descriptions</b>	configure Z-phase output start location and width
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>start_loc</b>	Z-phase output start location
<b>Input parameter{in}</b>	



<b>width</b>	Z-phase output width
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure Z-phase output start location and width */
edout_z_output_start_loc_and_width_config(92, 2);
```

## 3.12. EFUSE

The Efuse controller has efuse macro that store system paramters. As a non-volatile unit of storage, the bit of efuse macro cannot be restored to 0 once it is programmed to 1. The EFUSE registers are listed in chapter [3.12.1](#), the EFUSE firmware functions are introduced in chapter [3.12.2](#).

### 3.12.1. Descriptions of Peripheral registers

EFUSE registers are listed in the table shown as below:

**Table 3-333. EFUSE Registers**

Registers	Descriptions
EFUSE_CTL	EFUSE control register
EFUSE_ADDR	EFUSE address register
EFUSE_STAT	EFUSE status register
EFUSE_STATC	EFUSE status clear register
EFUSE_USER_CTL	EFUSE user control register
EFUSE_MCU_RSV	EFUSE MCU reserved register
EFUSE_DP <sub>x</sub> (x = 0,1)	EFUSE debug password register x(x = 0,1)
EFUSE_AES_KEY <sub>x</sub> (x = 0...3)	EFUSE firmware AES key register x(x = 0...3)
EFUSE_USER_DATA <sub>x</sub> (x = 0...3)	user data register x(x = 0...3)

### 3.12.2. Descriptions of Peripheral functions

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

**Table 3-334. EFUSE firmware function**

Function name	Function description
efuse_read	read system parameters from EFUSE macro to registers
efuse_write	program register values to EFUSE macro system parameters

Function name	Function description
efuse_user_control_write	program all user control parameters
efuse_mcu_reserved_write	program all MCU reserved parameters
efuse_dp_write	program all debug password parameters
efuse_aes_key_write	program all AES key parameters
efuse_user_data_write	program all user data parameters
efuse_aes_key_crc_get	get 8-bits CRC calculation result value of AES key
efuse_monitor_program_voltage_enable	enable monitor program voltage function
efuse_monitor_program_voltage_disable	disable monitor program voltage function
efuse_monitor_program_voltage_get	get monitor program voltage function
efuse_ldo_ready_get	get ldo ready signal
efuse_flag_get	check EFUSE flag is set or not
efuse_flag_clear	clear EFUSE pending flag
efuse_interrupt_enable	enable EFUSE interrupt
efuse_interrupt_disable	disable EFUSE interrupt
efuse_interrupt_flag_get	check EFUSE interrupt flag is set or not
efuse_interrupt_flag_clear	clear EFUSE pending interrupt flag

### Enum efuse\_system\_para\_size\_enum

Table 3-335. Enum efuse\_system\_para\_size\_enum

Member name	Function description
USER_CTL_MCU_RESERVED_SIZE	user control parameter size, MCU reserved parameter size
DP_SIZE	debug password parameter size
AES_KEY_SIZE	AES key parameter size
USER_DATA_SIZE	user data parameter size

### Enum efuse\_system\_para\_index\_enum

Table 3-336. Enum efuse\_system\_para\_index\_enum

Member name	Function description
USER_CTL_IDX	index of user control parameter
MCU_RESERVED_IDX	index of MCU reserved parameter
DP_IDX	index of debug password parameter
AES_KEY_IDX	index of AES key parameter
USER_DATA_IDX	index of user data parameter

## Enum efuse\_state\_enum

**Table 3-337. Enum efuse\_state\_enum**

Member name	Function description
EFUSE_READY	EFUSE operation has been completed
EFUSE_BUSY	EFUSE operation is in progress
EFUSE_IAERR	illegal access error
EFUSE_PVERR	program voltage setting error
EFUSE_TOERR	EFUSE timeout error

## Enum efuse\_interrupt\_flag\_enum

**Table 3-338. Enum efuse\_interrupt\_flag\_enum**

Member name	Function description
EFUSE_INT_FLAG_ILLEGAL_ACCESS_ERR	illegal access error interrupt flag
EFUSE_INT_FLAG_PROGRAM_COMPLETE	programming operation completion interrupt flag
EFUSE_INT_FLAG_READ_COMPLETE	read operation completion interrupt flag
EFUSE_INT_FLAG_PROGRAM_VOLTAGE_ERROR	program voltage setting error flag

## efuse\_read

The description of efuse\_read is shown as below:

**Table 3-339. Function efuse\_read**

Function name	efuse_read
Function prototype	ErrStatus efuse_read(uint32_t ef_addr, uint32_t size, uint32_t buf[]);
Function descriptions	read system parameters from EFUSE macro to registers
Precondition	-
The called functions	-
Input parameter{in}	
ef_addr	start address of the system parameters to be read
USER_CTL_EFADDR	user control parameter start address
MCU_RESERVED_EFADDR	MCU reserved parameter start address
DP_EFADDR	debug password parameter start address
USER_DATA_EFADDR	user data parameter start address
Input parameter{in}	
size	size of the system parameters to be read
USER_CTL_SIZE	user control parameter size
MCU_RESERVED_SIZE	MCU reserved parameter size

<i>DP_SIZE</i>	debug password parameter size
<i>USER_DATA_SIZE</i>	user data parameter size
<b>Input parameter{in}</b>	
<b>buf</b>	the buffer for data read from EFUSE macro
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* read user data from EFUSE macro to registers */
```

```
ErrStatus error_status = efuse_read(USER_DATA_IDX);
```

### efuse\_write

The description of efuse\_write is shown as below:

**Table 3-340. Function efuse\_program**

<b>Function name</b>	efuse_write
<b>Function prototype</b>	ErrStatus efuse_write(uint32_t ef_addr, uint32_t size, uint8_t *buf);
<b>Function descriptions</b>	program register values to EFUSE macro system parameters
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ef_addr</b>	the EFUSE address to be programmed, pgm_addr cannot exceed 384, and must be an integral multiple of 8
<b>Input parameter{in}</b>	
<b>size</b>	byte count to program
<b>Input parameter{in}</b>	
<b>buf</b>	the buffer for data written to EFUSE macro
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* write EFUSE USER DATA*/
```

```
uint8_t buffer[8] = { 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88 };
```

```
ErrStatus flag = ERROR;
```

```
flag = efuse_program (0x100, 8, buffer);
```

## efuse\_user\_control\_write

The description of efuse\_user\_control\_write is shown as below:

**Table 3-341. Function efuse\_user\_control\_write**

<b>Function name</b>	efuse_user_control_write
<b>Function prototype</b>	ErrStatus efuse_user_control_write(uint8_t *buf);
<b>Function descriptions</b>	program all user control parameters
<b>Precondition</b>	-
<b>The called functions</b>	efuse_program
<b>Input parameter{in}</b>	
<b>buf</b>	the buffer for data written to EFUSE macro
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* write EFUSE user control parameter */
uint8_t buffer[4] = { 0x11, 0x22, 0x33, 0x44 };
ErrStatus flag = ERROR;
flag = efuse_user_control_write(buffer);
```

## efuse\_mcu\_reserved\_write

The description of efuse\_mcu\_reserved\_write is shown as below:

**Table 3-342. Function efuse\_mcu\_reserved\_write**

<b>Function name</b>	efuse_mcu_reserved_write
<b>Function prototype</b>	ErrStatus efuse_mcu_reserved_write(uint8_t *buf);
<b>Function descriptions</b>	program all MCU reserved parameters
<b>Precondition</b>	-
<b>The called functions</b>	efuse_program
<b>Input parameter{in}</b>	
<b>buf</b>	the buffer for data written to EFUSE macro
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* write EFUSE MCU reserved parameter */
uint8_t buffer[4] = { 0x11, 0x22, 0x33, 0x44 };
```

```
ErrStatus flag = ERROR;
```

```
flag = efuse_mcu_reserved_write(buffer);
```

### efuse\_dp\_write

The description of efuse\_dp\_write is shown as below:

**Table 3-343. Function efuse\_dp\_write**

<b>Function name</b>	efuse_dp_write
<b>Function prototype</b>	ErrStatus efuse_dp_write(uint8_t *buf);
<b>Function descriptions</b>	program all debug password parameters
<b>Precondition</b>	-
<b>The called functions</b>	efuse_program
<b>Input parameter{in}</b>	
<b>buf</b>	the buffer for data written to EFUSE macro
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* write EFUSE debug password parameter */
```

```
uint8_t buffer[8] = { 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88 };
```

```
ErrStatus flag = ERROR;
```

```
flag = efuse_dp_write(buffer);
```

### efuse\_aes\_key\_write

The description of efuse\_aes\_key\_write is shown as below:

**Table 3-344. Function efuse\_aes\_key\_write**

<b>Function name</b>	efuse_aes_key_write
<b>Function prototype</b>	ErrStatus efuse_aes_key_write(uint8_t *buf);
<b>Function descriptions</b>	program all AES key parameters
<b>Precondition</b>	-
<b>The called functions</b>	efuse_program
<b>Input parameter{in}</b>	
<b>buf</b>	the buffer for data written to EFUSE macro
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* write EFUSE AES key parameter */
```

```
uint8_t buffer[16] = { 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB,
0xCC, 0xDD, 0xEE, 0xFF, 0x11 };
```

```
ErrStatus flag = ERROR;
```

```
flag = efuse_aes_key_write(buffer);
```

### efuse\_user\_data\_write

The description of efuse\_aes\_key\_write is shown as below:

**Table 3-345. Function efuse\_user\_data\_write**

<b>Function name</b>	efuse_user_data_write
<b>Function prototype</b>	ErrStatus efuse_user_data_write(uint8_t *buf);
<b>Function descriptions</b>	program all user data parameters
<b>Precondition</b>	-
<b>The called functions</b>	efuse_program
<b>Input parameter{in}</b>	
<b>buf</b>	the buffer for data written to EFUSE macro
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* write EFUSE user data parameter */
```

```
uint8_t buffer[16] = { 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB,
0xCC, 0xDD, 0xEE, 0xFF, 0x11 };
```

```
ErrStatus flag = ERROR;
```

```
flag = efuse_user_data_write(buffer);
```

### efuse\_aes\_key\_crc\_get

The description of efuse\_aes\_key\_crc\_get is shown as below:

**Table 3-346. Function efuse\_aes\_key\_crc\_get**

<b>Function name</b>	efuse_aes_key_crc_get
<b>Function prototype</b>	uint8_t efuse_aes_key_crc_get(void);
<b>Function descriptions</b>	get 8-bits CRC calculation result value of AES key
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-

Output parameter{out}	
-	-
Return value	
uint8_t	8-bits CRC calculation result value of AES key

Example:

```
/* get 8-bits CRC calculation result value of AES key */
```

```
uint8_t crc_value = 0;
```

```
crc_value = efuse_aes_key_crc_get();
```

### efuse\_monitor\_program\_voltage\_enable

The description of efuse\_monitor\_program\_voltage\_enable is shown as below:

**Table 3-347. Function efuse\_monitor\_program\_voltage\_enable**

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

Example:

```
/* enable monitor program voltage function */
```

```
efuse_monitor_program_voltage_enable();
```

### efuse\_monitor\_program\_voltage\_disable

The description of efuse\_monitor\_program\_voltage\_disable is shown as below:

**Table 3-348. Function efuse\_monitor\_program\_voltage\_disable**

Function name	efuse_monitor_program_voltage_disable
Function prototype	void efuse_monitor_program_voltage_disable(void);
Function descriptions	disable monitor program voltage function
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	



-	-
Return value	
-	-

Example:

```
/* disable monitor program voltage function */
```

```
efuse_monitor_program_voltage_disable();
```

### efuse\_monitor\_program\_voltage\_get

The description of efuse\_monitor\_program\_voltage\_get is shown as below:

**Table 3-349. Function efuse\_monitor\_program\_voltage\_get**

<b>Function name</b>	efuse_monitor_program_voltage_get
<b>Function prototype</b>	void efuse_monitor_program_voltage_get(void);
<b>Function descriptions</b>	get monitor program voltage function
<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 monitor program voltage function */
```

```
FlagStatus flag_sts = efuse_monitor_program_voltage_get();
```

### efuse\_ldo\_ready\_get

The description of efuse\_ldo\_ready\_get is shown as below:

**Table 3-350. Function efuse\_ldo\_ready\_get**

<b>Function name</b>	efuse_ldo_ready_get
<b>Function prototype</b>	FlagStatus efuse_ldo_ready_get(void);
<b>Function descriptions</b>	get ldo ready signal
<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 ldo ready signal */
```

```
FlagStatus flag_sts = efuse_ldo_ready_get();
```

### efuse\_flag\_get

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

**Table 3-351. Function efuse\_flag\_get**

<b>Function name</b>	efuse_flag_get
<b>Function prototype</b>	FlagStatus efuse_flag_get(uint32_t efuse_flag);
<b>Function descriptions</b>	check EFUSE flag is set or not
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>efuse_flag</b>	specifies to get a flag
EFUSE_FLAG_ILLEGAL_ACCESS_ERR	illegal access error flag
EFUSE_FLAG_PROGRAM_COMPLETE	programming operation completion flag
EFUSE_FLAG_READ_COMPLETE	read operation completion flag
EFUSE_FLAG_PROGRAM_VOLTAGE_ERR	program voltage setting error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the EFUSE illegal access error flag */
```

```
FlagStatus flag_value;
```

```
flag_value = efuse_flag_get(EFUSE_FLAG_ILLEGAL_ACCESS_ERR);
```

### efuse\_flag\_clear

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

**Table 3-352. Function efuse\_flag\_clear**

<b>Function name</b>	efuse_flag_clear
<b>Function prototype</b>	void efuse_flag_clear(uint32_t efuse_cflag);
<b>Function descriptions</b>	clear EFUSE pending flag

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>efuse_periph</b>	specifies to clear a flag
<i>EFUSE_FLAG_ILLEGAL_ACCESS_ERR_CLR</i>	illegal access error flag
<i>EFUSE_FLAG_PROGRAM_COMPLETE_CLR</i>	programming operation completion flag
<i>EFUSE_FLAG_READ_COMPLETE_CLR</i>	read operation completion flag
<i>EFUSE_FLAG_PROGRAM_VOLTAGE_ERR_CLR</i>	program voltage setting error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the EFUSE illegal access error flag */
```

```
efuse_flag_clear(EFUSE_FLAG_ILLEGAL_ACCESS_ERR_CLR);
```

### efuse\_interrupt\_enable

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

**Table 3-353. Function efuse\_interrupt\_enable**

<b>Function name</b>	efuse_interrupt_enable
<b>Function prototype</b>	void efuse_interrupt_enable(uint32_t source);
<b>Function descriptions</b>	enable EFUSE interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>source</b>	specifies an interrupt to enable
<i>EFUSE_INT_ILLEGAL_ACCESS_ERR</i>	illegal access error interrupt
<i>EFUSE_INT_PROGRAM_COMPLETE</i>	programming operation completion interrupt
<i>EFUSE_INT_READ_COMPLETE</i>	read operation completion interrupt
<i>EFUSE_INT_PROGRAM_VOLTAGE_ERR</i>	program voltage setting error interrupt
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* enable EFUSE illegal access error interrupt */
efuse_interrupt_enable(EFUSE_INT_ILLEGAL_ACCESS_ERR);
```

### efuse\_interrupt\_disable

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

**Table 3-354. Function efuse\_interrupt\_disable**

<b>Function name</b>	efuse_interrupt_disable
<b>Function prototype</b>	void efuse_interrupt_disable(uint32_t source);
<b>Function descriptions</b>	disable EFUSE interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>source</b>	specifies an interrupt to disable
EFUSE_INT_ILLEGAL_ACCESS_ERR	illegal access error interrupt
EFUSE_INT_PROGRAM_COMPLETE	programming operation completion interrupt
EFUSE_INT_READ_COMPLETE	read operation completion interrupt
EFUSE_INT_PROGRAM_VOLTAGE_ERR	program voltage setting error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable EFUSE illegal access error interrupt */
efuse_interrupt_disable(EFUSE_INT_ILLEGAL_ACCESS_ERR);
```

### efuse\_interrupt\_flag\_get

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

**Table 3-355. Function efuse\_interrupt\_flag\_get**

<b>Function name</b>	efuse_interrupt_flag_get
<b>Function prototype</b>	FlagStatus efuse_interrupt_flag_get(uint32_t int_flag);

<b>Function descriptions</b>	check EFUSE interrupt flag is set or not
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>efuse_flag</b>	specifies to get a flag
<i>EFUSE_INT_FLAG_ILLEGAL_ACCESS_ERR</i>	illegal access error interrupt flag
<i>EFUSE_INT_FLAG_PROGRAM_COMPLETE</i>	programming operation completion interrupt flag
<i>EFUSE_INT_FLAG_READ_COMPLETE</i>	read operation completion interrupt flag
<i>EFUSE_INT_FLAG_PROGRAM_VOLTAGE_ERROR</i>	program voltage setting error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the EFUSE illegal access error interrupt bits*/
```

```
FlagStatus flag_value;
```

```
flag_value = efuse_interrupt_flag_get(EFUSE_INT_FLAG_ILLEGAL_ACCESS_ERR);
```

### efuse\_interrupt\_flag\_clear

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

**Table 3-356. Function efuse\_interrupt\_flag\_clear**

<b>Function name</b>	efuse_interrupt_flag_clear
<b>Function prototype</b>	void efuse_interrupt_flag_clear(uint32_t efuse_periph, uint32_t int_flag);
<b>Function descriptions</b>	clear EFUSE pending interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>efuse_flag</b>	specifies to clear a flag
<i>EFUSE_INT_FLAG_ILLEGAL_ACCESS_ERR_CLR</i>	clear illegal access error interrupt flag
<i>EFUSE_INT_FLAG_PROGRAM_COMPLETE_CLR</i>	clear programming operation completion interrupt flag
<i>EFUSE_INT_FLAG_READ_COMPLETE_CLR</i>	clear operation completion interrupt flag

AD_COMPLETE_CLR	
EFUSE_INT_FLAG_PR OGRAM_VOLTAGE_E RR_CLR	clear program voltage setting error interrupt flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear the EFUSE illegal access error interrupt bits*/
```

```
efuse_interrupt_flag_clear(EFUSE_INT_FLAG_ILLEGAL_ACCESS_ERR_CLR);
```

## 3.13. EXMC

The external memory controller EXMC, is used as a translator for MCU to access a variety of external memory. The EXMC registers are listed in chapter [3.13.1](#), the EXMC firmware functions are introduced in chapter [3.13.2](#).

### 3.13.1. Descriptions of Peripheral registers

EXMC registers are listed in the table shown as below:

**Table 3-357. EXMC Registers**

Registers	Descriptions
EXMC_SNCTL	SRAM/NOR flash control registers
EXMC_SNTCFG	SRAM/NOR flash timing configuration registers
EXMC_SNWTCFG	SRAM/NOR flash write timing configuration registers
EXMC_NCTL	NAND flash control registers
EXMC_NINTEN	NAND flash interrupt enable registers
EXMC_NCTCFG	NAND flash common space timing configuration registers
EXMC_NATCFG	NAND flash attribute space timing configuration registers
EXMC_NECC	NAND flash ECC registers
EXMC_SDCTL	SDRAM control registers
EXMC_SDTCFG	SDRAM timing configuration registers
EXMC_SDCMD	SDRAM command register
EXMC_SDARI	SDRAM auto-refresh interval register
EXMC_SDSTAT	SDRAM status register
EXMC_SDRSCTL	SDRAM read sample control register

### 3.13.2. Descriptions of Peripheral functions

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

**Table 3-358. EXMC firmware function**

Function name	Function description
exmc_norsram_deinit	deinitialize EXMC NOR/SRAM regionx
exmc_norsram_struct_para_init	initialize exmc_norsram_parameter_struct with the default values
exmc_norsram_init	initialize EXMC NOR/SRAM regionx
exmc_norsram_enable	enable EXMC NOR/PSRAM regionx
exmc_norsram_disable	disable EXMC NOR/PSRAM regionx
exmc_nand_deinit	deinitialize EXMC NAND bankx
exmc_nand_struct_para_init	initialize exmc_norsram_parameter_struct with the default values
exmc_nand_init	initialize EXMC NAND bankx
exmc_nand_enable	enable EXMC NAND bankx
exmc_nand_disable	disable EXMC NAND bankx
exmc_sdram_deinit	deinitialize EXMC SDRAM devicex
exmc_sdram_struct_para_init	initialize exmc_sdram_parameter_struct with the default values
exmc_sdram_init	initialize EXMC SDRAM devicex
exmc_norsram_sdram_remap_config	configure NOR/PSRAM and SDRAM remap
exmc_norsram_sdram_remap_get	get NOR/PSRAM and SDRAM remap configuration
exmc_norsram_consecutive_clock_config	configure consecutive clock
exmc_norsram_page_size_config	configure CRAM page size
exmc_nand_ecc_config	enable or disable the EXMC NAND ECC function
exmc_ecc_get	get the EXMC ECC value
exmc_sdram_readsample_enable	enable read sample function
exmc_sdram_readsample_disable	disable read sample function
exmc_sdram_readsample_config	configure the delayed sample clock of read data
exmc_sdram_command_config	configure the SDRAM memory command
exmc_sdram_refresh_count_set	set auto-refresh interval
exmc_sdram_autorefresh_number_set	set the number of successive auto-refresh command
exmc_sdram_write_protection_config	config the write protection function
exmc_sdram_bankstatus_get	get the status of SDRAM device0 or device1
exmc_flag_get	get EXMC flag status
exmc_flag_clear	clear EXMC flag status
exmc_interrupt_enable	enable EXMC interrupt
exmc_interrupt_disable	disable EXMC interrupt
exmc_interrupt_flag_get	get EXMC interrupt flag
exmc_interrupt_flag_clear	clear EXMC interrupt flag

## Structure exmc\_norsram\_timing\_parameter\_struct

**Table 3-359. Structure exmc\_norsram\_timing\_parameter\_struct**

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

## Structure exmc\_norsram\_parameter\_struct

**Table 3-360. Structure exmc\_norsram\_parameter\_struct**

Member name	Function description
norsram_region	select the region of EXMC NOR/SRAM
write_mode	the write mode, synchronous mode or asynchronous mode
extended_mode	enable or disable the extended mode
asyn_wait	enable or disable the asynchronous wait function
nwait_signal	enable or disable the NWAIT signal while in synchronous bust mode
memory_write	enable or disable the write operation
nwait_config	NWAIT signal configuration
nwait_polarity	specifies the polarity of NWAIT signal from memory
burst_mode	enable or disable the burst mode
databus_width	specifies the databus width of external memory
memory_type	specifies the type of external memory
address_data_mux	specifies whether the data bus and address bus are multiplexed
cram_page_size	specifies CRAM page size
read_write_timing	timing parameters for read and write if the extended mode is not used or the timing parameters for read if the extended mode is used
write_timing	timing parameters for write when the extended mode is used

## Structure exmc\_nand\_timing\_parameter\_struct

**Table 3-361. Structure exmc\_nand\_timing\_parameter\_struct**

Member name	Function description
databus_hiztime	configure the databus HiZ time for write operation
holdtime	configure the address hold time(or the data hold time for write operation)
waittime	configure the minimum wait time
setup_time	configure the address setup time



## Structure exmc\_nand\_parameter\_struct

**Table 3-362. Structure exmc\_nand\_parameter\_struct**

Member name	Function description
ecc_size	the page size for the ECC calculation
atr_latency	configure the latency of ALE low to RB low
ctr_latency	configure the latency of CLE low to RB low
ecc_logic	enable or disable the ECC calculation logic
databus_width	the NAND flash databus width
wait_feature	enable or disable the wait feature
common_space_timing	the timing parameters for NAND flash common space
attribute_space_timing	the timing parameters for NAND flash attribute space

## Structure exmc\_sdram\_timing\_parameter\_struct

**Table 3-363. Structure exmc\_sdram\_timing\_parameter\_struct**

Member name	Function description
row_to_column_delay	configure the row to column delay
row_precharge_delay	configure the row precharge delay
write_recovery_delay	configure the write recovery delay
auto_refresh_delay	configure the auto refresh delay
row_address_select_delay	configure the row address select delay
exit_selfrefresh_delay	configure the exit self-refresh delay
load_mode_register_delay	configure the load mode register delay

## Structure exmc\_sdram\_parameter\_struct

**Table 3-364. Structure exmc\_sdram\_parameter\_struct**

Member name	Function description
sdram_device	select the device of SDRAM
pipeline_read_delay	the delay for reading data after CAS latency in CK_EXMC clock cycles
brust_read_switch	enable or disable the burst read
sdclk_config	the SDCLK memory clock for both SDRAM banks
write_protection	enable or disable SDRAM bank write protection function
cas_latency	configure the SDRAM CAS latency
internal_bank_numb	the number of internal bank

Member name	Function description
er	
data_width	the databus width of SDRAM memory
row_address_width	the bit width of a row address
column_address_width	the bit width of a column address
timing	the timing parameters for write and read SDRAM

### Structure `exmc_sdram_command_parameter_struct`

**Table 3-365. Structure `exmc_sdram_command_parameter_struct`**

Member name	Function description
mode_register_content	the SDRAM mode register content
auto_refresh_number	the number of successive auto-refresh cycles will be send when CMD = 011
bank_select	the bank which command will be sent to
command	the commands that will be sent to SDRAM

### `exmc_norsram_deinit`

The description of `exmc_norsram_deinit` is shown as below:

**Table 3-366. Function `exmc_norsram_deinit`**

<b>Function name</b>	<code>exmc_norsram_deinit</code>
<b>Function prototype</b>	<code>void exmc_norsram_deinit(uint32_t exmc_norsram_region);</code>
<b>Function descriptions</b>	deinitialize EXMC NOR/SRAM region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><code>exmc_norsram_region</code></b>	EXMC NOR/SRAM region
<b><code>EXMC_BANK0_NORSRAM_REGIONx</code></b>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize the EXMC NOR/SRAM region1 of bank0 */
exmc_norsram_deinit(EXMC_BANK0_NORSRAM_REGION1);
```

## exmc\_norsram\_struct\_para\_init

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

**Table 3-367. Function exmc\_norsram\_struct\_para\_init**

<b>Function name</b>	exmc_norsram_struct_para_init
<b>Function prototype</b>	void exmc_norsram_struct_para_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
<b>Function descriptions</b>	initialize the struct exmc_norsram_parameter_struct
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>exmc_norsram_init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-360. Structure exmc_norsram_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize the struct nor_init_struct */
exmc_norsram_parameter_struct nor_init_struct;
exmc_norsram_struct_para_init(&nor_init_struct);
```

## exmc\_norsram\_init

The description of exmc\_norsram\_init is shown as below:

**Table 3-368. Function exmc\_norsram\_init**

<b>Function name</b>	exmc_norsram_init
<b>Function prototype</b>	void exmc_norsram_init(exmc_norsram_parameter_struct* exmc_norsram_init_struct);
<b>Function descriptions</b>	initialize EXMC NOR/SRAM region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>exmc_norsram_init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-360. Structure exmc_norsram_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize EXMC NOR/SRAM bank */

exmc_norsram_parameter_struct lcd_init_struct;

exmc_norsram_timing_parameter_struct lcd_timing_init_struct;

exmc_norsram_struct_para_init(&lcd_init_struct);

/* configure timing parameter */

lcd_timing_init_struct.asyn_access_mode = EXMC_ACCESS_MODE_A;

lcd_timing_init_struct.syn_data_latency = EXMC_DATA_LAT_2_CLK;

lcd_timing_init_struct.syn_clk_division = EXMC_SYN_CLOCK_RATIO_DISABLE;

lcd_timing_init_struct.bus_latency = 1;

lcd_timing_init_struct.asyn_data_setup_time = 5;

lcd_timing_init_struct.asyn_address_hold_time = 2;

lcd_timing_init_struct.asyn_address_setup_time = 2;

/* configure EXMC bus parameters */

lcd_init_struct.norsram_region = EXMC_BANK0_NORSRAM_REGION1;

lcd_init_struct.write_mode = EXMC_ASYN_WRITE;

lcd_init_struct.extended_mode = DISABLE;

lcd_init_struct.asyn_wait = DISABLE;

lcd_init_struct.nwait_signal = DISABLE;

lcd_init_struct.memory_write = ENABLE;

lcd_init_struct.nwait_config = EXMC_NWAIT_CONFIG_BEFORE;

lcd_init_struct.nwait_polarity = EXMC_NWAIT_POLARITY_LOW;

lcd_init_struct.burst_mode = DISABLE;

lcd_init_struct.databus_width = EXMC_NOR_DATABUS_WIDTH_16B;

lcd_init_struct.memory_type = EXMC_MEMORY_TYPE_SRAM;

lcd_init_struct.address_data_mux = DISABLE;

lcd_init_struct.cram_page_size = EXMC_CRAM_AUTO_SPLIT;

lcd_init_struct.read_write_timing = &lcd_timing_init_struct;

lcd_init_struct.write_timing = &lcd_timing_init_struct;

exmc_norsram_init(&lcd_init_struct);
```

## exmc\_norsram\_enable

The description of exmc\_norsram\_enable is shown as below:

**Table 3-369. Function exmc\_norsram\_enable**

<b>Function name</b>	exmc_norsram_enable
<b>Function prototype</b>	void exmc_norsram_enable(uint32_t exmc_norsram_region);
<b>Function descriptions</b>	enable EXMC NOR/PSRAM region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_norsram_region</b>	EXMC NOR/SRAM region
<i>EXMC_BANK0_NORSRAM_REGIONx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the EXMC NOR/SRAM region1 of bank0 */
exmc_norsram_enable(EXMC_BANK0_NORSRAM_REGION1);
```

## exmc\_norsram\_disable

The description of exmc\_norsram\_disable is shown as below:

**Table 3-370. Function exmc\_norsram\_disable**

<b>Function name</b>	exmc_norsram_disable
<b>Function prototype</b>	void exmc_norsram_disable(uint32_t exmc_norsram_region);
<b>Function descriptions</b>	disable EXMC NOR/PSRAM region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_norsram_region</b>	EXMC NOR/SRAM region
<i>EXMC_BANK0_NORSRAM_REGIONx</i>	x=0,1,2,3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the EXMC NOR/SRAM region1 of bank0 */
exmc_norsram_disable(EXMC_BANK0_NORSRAM_REGION1);
```

### exmc\_nand\_deinit

The description of exmc\_nand\_deinit is shown as below:

**Table 3-371. Function exmc\_nand\_deinit**

<b>Function name</b>	exmc_nand_deinit
<b>Function prototype</b>	void exmc_nand_deinit(void);
<b>Function descriptions</b>	deinitialize EXMC NAND bank
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize EXMC NAND bank */
exmc_nand_deinit();
```

### exmc\_nand\_struct\_para\_init

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

**Table 3-372. Function exmc\_nand\_struct\_para\_init**

<b>Function name</b>	exmc_nand_struct_para_init
<b>Function prototype</b>	void exmc_nand_struct_para_init(exmc_nand_parameter_struct* exmc_nand_init_struct);
<b>Function descriptions</b>	initialize the struct exmc_nand_parameter_struct
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>exmc_nand_init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-362</a> . <a href="#">Structure exmc_nand_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

```

/* initialize the struct nand_init_struct */

exmc_nand_parameter_struct nand_init_struct;

exmc_nand_struct_para_init (&nand_init_struct);

```

## exmc\_nand\_init

The description of exmc\_nand\_init is shown as below:

**Table 3-373. Function exmc\_nand\_init**

<b>Function name</b>	exmc_nand_init
<b>Function prototype</b>	void exmc_nand_init(exmc_nand_parameter_struct* exmc_nand_init_struct);
<b>Function descriptions</b>	initialize EXMC NAND bank
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_nand_init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-362. Structure exmc_nand_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

exmc_nand_parameter_struct nand_init_struct;

exmc_nand_timing_parameter_struct nand_timing_init_struct;

exmc_nand_struct_para_init (&nand_init_struct);

/* EXMC configuration */

nand_timing_init_struct.setuptime = 5;

nand_timing_init_struct.waittime = 4;

nand_timing_init_struct.holdtime = 2;

nand_timing_init_struct.databus_hiztime = 2;

nand_init_struct.ecc_size = EXMC_ECC_SIZE_2048BYTES;

nand_init_struct.atr_latency = EXMC_ALE_RE_DELAY_1_CK_EXMC;

nand_init_struct.ctr_latency = EXMC_CLE_RE_DELAY_1_CK_EXMC;

nand_init_struct.ecc_logic = ENABLE;

nand_init_struct.databus_width = EXMC_NAND_DATABUS_WIDTH_8B;

```

```

nand_init_struct.wait_feature = ENABLE;

nand_init_struct.common_space_timing = &nand_timing_init_struct;

nand_init_struct.attribute_space_timing = &nand_timing_init_struct;

exmc_nand_init(&nand_init_struct);

```

### exmc\_nand\_enable

The description of exmc\_nand\_enable is shown as below:

**Table 3-374. Function exmc\_nand\_enable**

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

Example:

```

/* enable EXMC NAND bank2 */

exmc_nand_enable();

```

### exmc\_nand\_disable

The description of exmc\_nand\_disable is shown as below:

**Table 3-375. Function exmc\_nand\_disable**

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

Example:



```
/* disable EXMC NAND bank2 */
```

```
exmc_nand_disable();
```

### exmc\_sdram\_deinit

The description of exmc\_sdram\_deinit is shown as below:

**Table 3-376. Function exmc\_sdram\_deinit**

<b>Function name</b>	exmc_sdram_deinit
<b>Function prototype</b>	void exmc_sdram_deinit(uint32_t exmc_sdram_device);
<b>Function descriptions</b>	deinitialize EXMC SDRAM device
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_sdram_device</b>	EXMC SDRAM device
<i>EXMC_SDRAM_DEVICE</i> <i>Ex</i>	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize EXMC SDRAM device1 */
```

```
exmc_sdram_deinit(EXMC_SDRAM_DEVICE1);
```

### exmc\_sdram\_struct\_para\_init

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

**Table 3-377. Function exmc\_sdram\_struct\_para\_init**

<b>Function name</b>	exmc_sdram_struct_para_init
<b>Function prototype</b>	exmc_sdram_struct_para_init(exmc_sdram_parameter_struct* exmc_sdram_init_struct);
<b>Function descriptions</b>	initialize the struct exmc_sdram_parameter_struct
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>exmc_sdram_init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-364. Structure exmc_sdram_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize the struct sdram_init_struct */
exmc_sdram_parameter_struct sdram_init_struct;
exmc_sdram_struct_para_init (&sdram_init_struct);
```

### exmc\_sdram\_init

The description of exmc\_sdram\_init is shown as below:

**Table 3-378. Function exmc\_sdram\_init**

Function name	exmc_sdram_init
Function prototype	void exmc_sdram_init(exmc_sdram_parameter_struct* exmc_sdram_init_struct);
Function descriptions	initialize EXMC SDRAM device
Precondition	-
The called functions	-
Input parameter{in}	
exmc_sdram_init_struct	Structure for initialization, the structure members can refer to <a href="#">Table 3-364. Structure exmc_sdram_parameter_struct</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
exmc_sdram_parameter_struct      sdram_init_struct;

exmc_sdram_timing_parameter_struct sdram_timing_init_struct;

/* EXMC configuration */

sdram_timing_init_struct.load_mode_register_delay = 2;

/* XSRD: min = 67ns */

sdram_timing_init_struct.exit_selfrefresh_delay = 7;

/* RASD: min=42ns , max=120k (ns) */

sdram_timing_init_struct.row_address_select_delay = 5;

/* ARFD: min=60ns */

sdram_timing_init_struct.auto_refresh_delay = 6;

/* WRD: min=1 Clock cycles +6ns */

sdram_timing_init_struct.write_recovery_delay = 2;
```

```

/* RPD:  min=18ns */

sdram_timing_init_struct.row_precharge_delay = 2;

/* RCD:  min=18ns */

sdram_timing_init_struct.row_to_column_delay = 2;

sdram_init_struct.sdram_device = sdram_device;

sdram_init_struct.column_address_width = EXMC_SDRAM_COW_ADDRESS_9;

sdram_init_struct.row_address_width = EXMC_SDRAM_ROW_ADDRESS_13;

sdram_init_struct.data_width = EXMC_SDRAM_DATABUS_WIDTH_16B;

sdram_init_struct.internal_bank_number = EXMC_SDRAM_4_INTER_BANK;

sdram_init_struct.cas_latency = EXMC_CAS_LATENCY_3_SDCLK;

sdram_init_struct.write_protection = DISABLE;

sdram_init_struct.sdclk_config = EXMC_SDCLK_PERIODS_2_CK_EXMC;

sdram_init_struct.burst_read_switch = ENABLE;

sdram_init_struct.pipeline_read_delay = EXMC_PIPELINE_DELAY_1_CK_EXMC;

sdram_init_struct.timing = &sdram_timing_init_struct;

/* EXMC SDRAM bank initialization */

exmc_sdram_init(&sdram_init_struct);

```

### exmc\_norsram\_sdram\_remap\_config

The description of exmc\_norsram\_sdram\_remap\_config is shown as below:

**Table 3-379. Function exmc\_norsram\_sdram\_remap\_config**

Function name	exmc_norsram_sdram_remap_config
Function prototype	void exmc_norsram_sdram_remap_config(uint32_t bank_remap);
Function descriptions	configure NOR/PSRAM and SDRAM remap
Precondition	-
The called functions	-
Input parameter{in}	
bank_remap	NOR/PSRAM and SDRAM map address
EXMC_BANK_REMAP_DEFAULT	default mapping
EXMC_BANK_NORPSRAM_SDRAM_SWAP	NOR/PSRAM bank and SDRAM device 0 swapped
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* configure NOR/PSRAM and SDRAM remap */
```

```
exmc_norsram_sdram_remap_config(EXMC_BANK_NORPSRAM_SDRAM_SWAP);
```

### exmc\_norsram\_sdram\_remap\_get

The description of exmc\_norsram\_sdram\_remap\_get is shown as below:

**Table 3-380. Function exmc\_norsram\_sdram\_remap\_get**

<b>Function name</b>	exmc_norsram_sdram_remap_get
<b>Function prototype</b>	uint32_t exmc_norsram_sdram_remap_get(void);
<b>Function descriptions</b>	get NOR/PSRAM and SDRAM remap configuration
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	EXMC_BANK_REMAP_DEFAULT, EXMC_BANK_NORPSRAM_SDRAM_SWAP

Example:

```
/* get NOR/PSRAM and SDRAM remap configuration */
```

```
uint32_t remap_state;
```

```
remap_state = exmc_norsram_sdram_remap_get();
```

### exmc\_norsram\_consecutive\_clock\_config

The description of exmc\_norsram\_consecutive\_clock\_config is shown as below:

**Table 3-381. Function exmc\_norsram\_consecutive\_clock\_config**

<b>Function name</b>	exmc_norsram_consecutive_clock_config
<b>Function prototype</b>	void exmc_norsram_consecutive_clock_config(uint32_t clock_mode);
<b>Function descriptions</b>	configure consecutive clock mode (consecutive clock is only supported in EXMC BANK0 REGION0)
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
clock_mode	the mode of consecutive clock
EXMC_CLOCK_SYN_MODE	the clock is generated only during synchronous access

<i>EXMC_CLOCK_UNCONDITIONALLY</i>	the clock is generated unconditionally
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure consecutive clock */
```

```
exmc_norsram_consecutive_clock_config(EXMC_CLOCK_SYN_MODE);
```

### exmc\_norsram\_page\_size\_config

The description of exmc\_norsram\_page\_size\_config is shown as below:

**Table 3-382. Function exmc\_norsram\_page\_size\_config**

<b>Function name</b>	exmc_norsram_page_size_config
<b>Function prototype</b>	void exmc_norsram_page_size_config(uint32_t page_size);
<b>Function descriptions</b>	configure CRAM page size
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>page_size</b>	CRAM page size
<i>EXMC_CRAM_AUTO_SPLIT</i>	the clock is generated only during synchronous access
<i>EXMC_CRAM_PAGE_SIZE_128_BYTES</i>	page size is 128 bytes
<i>EXMC_CRAM_PAGE_SIZE_256_BYTES</i>	page size is 256 bytes
<i>EXMC_CRAM_PAGE_SIZE_512_BYTES</i>	page size is 512 bytes
<i>EXMC_CRAM_PAGE_SIZE_1024_BYTES</i>	page size is 1024 bytes
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CRAM page size */
```

```
exmc_norsram_page_size_config (EXMC_CRAM_PAGE_SIZE_128_BYTES);
```

## exmc\_nand\_ecc\_config

The description of exmc\_nand\_ecc\_config is shown as below:

**Table 3-383. Function exmc\_nand\_ecc\_config**

<b>Function name</b>	exmc_nand_ecc_config
<b>Function prototype</b>	void exmc_nand_ecc_config(ControlStatus newvalue);
<b>Function descriptions</b>	enable or disable the EXMC NAND ECC function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>newvalue</b>	ENABLE or DISABLE
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the EXMC NAND ECC function */
exmc_nand_ecc_config(ENABLE);
```

## exmc\_ecc\_get

The description of exmc\_ecc\_get is shown as below:

**Table 3-384. Function exmc\_ecc\_get**

<b>Function name</b>	exmc_ecc_get
<b>Function prototype</b>	uint32_t exmc_ecc_get(void);
<b>Function descriptions</b>	get the EXMC ECC value
<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>	the error correction code(ECC) value

Example:

```
/* get the EXMC ECC value */
uint32_t ecc_value;
ecc_value = exmc_ecc_get();
```

## exmc\_sdram\_readsample\_enable

The description of exmc\_sdram\_readsample\_enable is shown as below:

**Table 3-385. Function exmc\_sdram\_readsample\_enable**

<b>Function name</b>	exmc_sdram_readsample_enable
<b>Function prototype</b>	void exmc_sdram_readsample_enable(void);
<b>Function descriptions</b>	enable read sample 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 read sample */
exmc_sdram_readsample_enable();
```

## exmc\_sdram\_readsample\_disable

The description of exmc\_sdram\_readsample\_disable is shown as below:

**Table 3-386. Function exmc\_sdram\_readsample\_disable**

<b>Function name</b>	exmc_sdram_readsample_disable
<b>Function prototype</b>	void exmc_sdram_readsample_disable(void);
<b>Function descriptions</b>	disable read sample 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 read sample */
exmc_sdram_readsample_disable();
```

## exmc\_sdram\_readsample\_config

The description of exmc\_sdram\_readsample\_config is shown as below:

Table 3-387. Function `exmc_sdram_readsample_config`

Function name	<code>exmc_sdram_readsample_config</code>
Function prototype	<code>void exmc_sdram_readsample_config(uint32_t delay_cell, uint32_t extra_clk);</code>
Function descriptions	configure the delayed sample clock of read data
Precondition	-
The called functions	-
Input parameter{in}	
<code>delay_cell</code>	SDRAM the delayed sample clock of read data
<code>EXMC_SDRAM_x_DELAY_CELL</code>	x=0...15
Input parameter{in}	
<code>extra_clk</code>	sample cycle of read data
<code>EXMC_SDRAM_READ_SAMPLE_x_EXTRACK</code>	x=0,1
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the delayed sample clock and sample cycle of read data */
exmc_sdram_readsample_config(EXMC_SDRAM_1_DELAY_CELL,
EXMC_SDRAM_READSAMPLE_1_EXTRACK);
```

### `exmc_sdram_command_config`

The description of `exmc_sdram_command_config` is shown as below:

Table 3-388. Function `exmc_sdram_command_config`

Function name	<code>exmc_sdram_command_config</code>
Function prototype	<code>void exmc_sdram_command_config(exmc_sdram_command_parameter_struct* exmc_sdram_command_init_struct);</code>
Function descriptions	configure the SDRAM memory command
Precondition	-
The called functions	-
Input parameter{in}	
<code>exmc_sdram_command_init_struct</code>	Structure for initialization, the structure members can refer to <a href="#">Table 3-365. Structure <code>exmc_sdram_command_parameter_struct</code></a>
Output parameter{out}	
-	-
Return value	
-	-



Example:

```
/* configure the SDRAM memory command */

exmc_sdram_command_parameter_struct      sdram_command_init_struct;

sdram_command_init_struct.command = EXMC_SDRAM_CLOCK_ENABLE;

sdram_command_init_struct.bank_select = bank_select;

sdram_command_init_struct.auto_refresh_number =
EXMC_SDRAM_AUTO_REFLESH_1_SDCLK;

sdram_command_init_struct.mode_register_content = 0;

exmc_sdram_command_config(&sdram_command_init_struct);
```

### exmc\_sdram\_refresh\_count\_set

The description of exmc\_sdram\_refresh\_count\_set is shown as below:

**Table 3-389. Function exmc\_sdram\_refresh\_count\_set**

<b>Function name</b>	exmc_sdram_refresh_count_set
<b>Function prototype</b>	void exmc_sdram_refresh_count_set(uint32_t exmc_count);
<b>Function descriptions</b>	set auto-refresh interval
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_count</b>	the number SDRAM clock cycles unit between two successive auto-refresh commands, 0x0000~0x1FFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the SDRAM auto-refresh rate counter */

exmc_sdram_refresh_count_set(761);
```

### exmc\_sdram\_autorefresh\_number\_set

The description of exmc\_sdram\_autorefresh\_number\_set is shown as below:

**Table 3-390. Function exmc\_sdram\_autorefresh\_number\_set**

<b>Function name</b>	exmc_sdram_autorefresh_number_set
<b>Function prototype</b>	void exmc_sdram_autorefresh_number_set(uint32_t exmc_number);
<b>Function descriptions</b>	set the number of successive auto-refresh command
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
exmc_number	the number of successive Auto-refresh cycles will be send
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set the number of successive auto-refresh command */
```

```
exmc_sdram_autorefresh_number_set(10);
```

### exmc\_sdram\_write\_protection\_config

The description of exmc\_sdram\_write\_protection\_config is shown as below:

**Table 3-391. Function exmc\_sdram\_write\_protection\_config**

Function name	exmc_sdram_write_protection_config
Function prototype	void exmc_sdram_write_protection_config(uint32_t exmc_sdram_device, ControlStatus newvalue);
Function descriptions	config the write protection function
Precondition	-
The called functions	-
Input parameter{in}	
exmc_sdram_device	specifie the SDRAM device
EXMC_SDRAM_DEVIC Ex	x=0,1
Input parameter{in}	
newvalue	ENABLE or DISABLE
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the write protection function */
```

```
exmc_sdram_write_protection_config(EXMC_SDRAM_DEVICE1, ENABLE);
```

### exmc\_sdram\_bankstatus\_get

The description of exmc\_sdram\_bankstatus\_get is shown as below:

**Table 3-392. Function exmc\_sdram\_bankstatus\_get**

Function name	exmc_sdram_bankstatus_get
---------------	---------------------------

<b>Function prototype</b>	uint32_t exmc_sdram_bankstatus_get(uint32_t exmc_sdram_device);
<b>Function descriptions</b>	get the status of SDRAM device0 or device1
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_sdram_device</b>	specifie the SDRAM device
<i>EXMC_SDRAM_DEVICE</i> Ex	x=0,1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the status of SDRAM device

Example:

```
/* get the status of SDRAM device1 */
```

```
uint32_t status;
```

```
status = exmc_sdram_bankstatus_get (EXMC_SDRAM_DEVICE1);
```

### exmc\_flag\_get

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

**Table 3-393. Function exmc\_flag\_get**

<b>Function name</b>	exmc_flag_get
<b>Function prototype</b>	FlagStatus exmc_flag_get(uint32_t exmc_bank,uint32_t flag);
<b>Function descriptions</b>	get EXMC flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_bank</b>	specifies the NAND bank or SDRAM device
<i>EXMC_BANK2_NAND</i>	the NAND bank2
<i>EXMC_SDRAM_DEVICE0</i>	the SDRAM device0
<i>EXMC_SDRAM_DEVICE1</i>	the SDRAM device1
<b>Input parameter{in}</b>	
<b>flag</b>	specify get which flag
<i>EXMC_NAND_FLAG_LEVEL</i>	interrupt high-level status
<i>EXMC_NAND_FLAG_RISE</i>	interrupt rising edge status
<i>EXMC_NAND_FLAG_FALL</i>	interrupt falling edge status

<i>EXMC_SDRAM_FLAG_REFRESH</i>	refresh error interrupt flag
<i>EXMC_SDRAM_FLAG_NREADY</i>	not ready status
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* check EXMC_NAND_FLAG_RISE is set or not */
```

```
if(RESET != exmc_flag_get (EXMC_BANK2_NAND, EXMC_NAND_FLAG_RISE));
```

### exmc\_flag\_clear

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

**Table 3-394. Function exmc\_flag\_clear**

<b>Function name</b>	exmc_flag_clear
<b>Function prototype</b>	FlagStatus exmc_flag_clear (uint32_t exmc_bank,uint32_t flag);
<b>Function descriptions</b>	clear EXMC flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_bank</b>	specifies the NAND bank or SDRAM device
<i>EXMC_BANK2_NAND</i>	the NAND bank2
<i>EXMC_SDRAM_DEVICE0</i>	the SDRAM device0
<i>EXMC_SDRAM_DEVICE1</i>	the SDRAM device1
<b>Input parameter{in}</b>	
<b>flag</b>	specify get which flag
<i>EXMC_NAND_FLAG_LEVEL</i>	interrupt high-level status
<i>EXMC_NAND_FLAG_RISE</i>	interrupt rising edge status
<i>EXMC_NAND_FLAG_FALL</i>	interrupt falling edge status
<i>EXMC_SDRAM_FLAG_REFRESH</i>	refresh error interrupt flag
<i>EXMC_SDRAM_FLAG_NREADY</i>	not ready status
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* clear EXMC flag status */
```

```
exmc_flag_clear(EXMC_BANK2_NAND, EXMC_NAND_FLAG_RISE);
```

### exmc\_interrupt\_enable

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

**Table 3-395. Function exmc\_interrupt\_enable**

Function name	exmc_interrupt_enable
Function prototype	void exmc_interrupt_enable(uint32_t exmc_bank, uint32_t interrupt);
Function descriptions	enable EXMC interrupt
Precondition	-
The called functions	-
Input parameter{in}	
<b>exmc_bank</b>	specifies the NAND bank or SDRAM device
<i>EXMC_BANK2_NAND</i>	the NAND bank2
<i>EXMC_SDRAM_DEVICE0</i>	the SDRAM device0
<i>EXMC_SDRAM_DEVICE1</i>	the SDRAM device1
Input parameter{in}	
<b>interrupt</b>	interrupt
<i>EXMC_NAND_INT_FLAG_LEVEL</i>	high-level interrupt
<i>EXMC_NAND_INT_FLAG_RISE</i>	rising edge interrupt
<i>EXMC_NAND_INT_FLAG_FALL</i>	falling edge interrupt
<i>EXMC_SDRAM_INT_FLAG_REFRESH</i>	refresh error interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable EXMC interrupt */
```

```
exmc_interrupt_enable(EXMC_BANK2_NAND, EXMC_NAND_INT_FLAG_RISE);
```

## exmc\_interrupt\_disable

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

**Table 3-396. Function exmc\_interrupt\_disable**

<b>Function name</b>	exmc_interrupt_disable
<b>Function prototype</b>	void exmc_interrupt_disable(uint32_t exmc_bank,uint32_t interrupt);
<b>Function descriptions</b>	disable EXMC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_bank</b>	specifies the NAND bank or SDRAM device
EXMC_BANK2_NAND	the NAND bank2
EXMC_SDRAM_DEVICE0	the SDRAM device0
EXMC_SDRAM_DEVICE1	the SDRAM device1
<b>Input parameter{in}</b>	
<b>interrupt</b>	interrupt
EXMC_NAND_INT_FLAG_LEVEL	high-level interrupt
EXMC_NAND_INT_FLAG_RISE	rising edge interrupt
EXMC_NAND_INT_FLAG_FALL	falling edge interrupt
EXMC_SDRAM_INT_FLAG_REFRESH	refresh error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable EXMC interrupt */
exmc_interrupt_disable(EXMC_BANK2_NAND, EXMC_NAND_INT_FLAG_RISE);
```

## exmc\_interrupt\_flag\_get

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

**Table 3-397. Function exmc\_interrupt\_flag\_get**

<b>Function name</b>	exmc_interrupt_flag_get
<b>Function prototype</b>	FlagStatus exmc_interrupt_flag_get(uint32_t exmc_bank,uint32_t interrupt);
<b>Function descriptions</b>	get EXMC interrupt flag

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_bank</b>	specifies the NAND bank or SDRAM device
<i>EXMC_BANK2_NAND</i>	the NAND bank2
<i>EXMC_SDRAM_DEVICE0</i>	the SDRAM device0
<i>EXMC_SDRAM_DEVICE1</i>	the SDRAM device1
<b>Input parameter{in}</b>	
<b>interrupt</b>	specify get which interrupt flag
<i>EXMC_NAND_INT_FLAG_LEVEL</i>	high-level interrupt and flag
<i>EXMC_NAND_INT_FLAG_RISE</i>	rising edge interrupt and flag
<i>EXMC_NAND_INT_FLAG_FALL</i>	falling edge interrupt and flag
<i>EXMC_SDRAM_INT_FLAG_REFRESH</i>	refresh error interrupt and flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* check EXMC_NAND INT_FLAG_RISE is set or not*/
```

```
if(RESET != exmc_interrupt_flag_get (EXMC_BANK2_NAND,
EXMC_NAND_INT_FLAG_RISE));
```

### exmc\_interrupt\_flag\_clear

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

**Table 3-398. Function exmc\_interrupt\_flag\_clear**

<b>Function name</b>	exmc_interrupt_flag_clear
<b>Function prototype</b>	void exmc_interrupt_flag_clear(uint32_t exmc_bank,uint32_t interrupt);
<b>Function descriptions</b>	clear EXMC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exmc_bank</b>	specifies the NAND bank or SDRAM device
<i>EXMC_BANK2_NAND</i>	the NAND bank2
<i>EXMC_SDRAM_DEVICE0</i>	the SDRAM device0

EXMC_SDRAM_DEVIC E1	the SDRAM device1
<b>Input parameter{in}</b>	
<b>interrupt</b>	specify get which interrupt flag
EXMC_NAND_INT_FL AG_LEVEL	high-level interrupt and flag
EXMC_NAND_INT_FL AG_RISE	rising edge interrupt and flag
EXMC_NAND_INT_FL AG_FALL	falling edge interrupt and flag
EXMC_SDRAM_INT_F LAG_REFRESH	refresh error interrupt and flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear EXMC interrupt flag */
exmc_interrupt_flag_clear(EXMC_BANK2_NAND, EXMC_NAND_INT_FLAG_RISE);
```

## 3.14. EXTI

EXTI is the interrupt/event controller in the MCU. It contains up to 38 independent edge detectors and generates interrupt requests or events to the processor. The EXTI registers are listed in chapter [3.14.1](#), the EXTI firmware functions are introduced in chapter [3.14.2](#).

### 3.14.1. Descriptions of Peripheral registers

EXTI registers are listed in the table shown as below:

**Table 3-399. EXTI Registers**

Registers	Descriptions
EXTI_INTEN0	interrupt enable register 0
EXTI_EVEN0	event enable register 0
EXTI_RTEN0	rising edge trigger enable register 0
EXTI_FTEN0	falling edge trigger enable register 0
EXTI_SWIEV0	software interrupt event register 0
EXTI_PD0	pending register 0
EXTI_INTEN1	interrupt enable register 1
EXTI_EVEN1	event enable register 1
EXTI_RTEN1	rising edge trigger enable register 1
EXTI_FTEN1	falling edge trigger enable register 1



Registers	Descriptions
EXTI_SWIEV1	software interrupt event register 1
EXTI_PD1	pending register 1

### 3.14.2. Descriptions of Peripheral functions

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

**Table 3-400. EXTI firmware function**

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

#### Enum exti\_line\_enum

**Table 3-401. Enum exti\_line\_enum**

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

enum name	Function description
EXTI_17	EXTI line 17
EXTI_18	EXTI line 18
EXTI_19	EXTI line 19
EXTI_20	EXTI line 20
EXTI_21	EXTI line 21
EXTI_22	EXTI line 22
EXTI_23	EXTI line 23
EXTI_24	EXTI line 24
EXTI_25	EXTI line 25
EXTI_26	EXTI line 26
EXTI_27	EXTI line 27
EXTI_28	EXTI line 28
EXTI_29	EXTI line 29
EXTI_30	EXTI line 30
EXTI_31	EXTI line 31
EXTI_32	EXTI line 32
EXTI_33	EXTI line 33
EXTI_34	EXTI line 34
EXTI_35	EXTI line 35
EXTI_36	EXTI line 36
EXTI_37	EXTI line 37

### Enum exti\_mode\_enum

Table 3-402. Enum exti\_mode\_enum

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

### Enum exti\_trig\_type\_enum

Table 3-403. Enum exti\_trig\_type\_enum

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

### exti\_deinit

The description of exti\_deinit is shown as below:

Table 3-404. 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 3-405. 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 3-401. Enum exti_line_enum</a>
Input parameter{in}	
mode	EXTI mode, refer to <a href="#">Table 3-402. Enum exti_mode_enum</a>
Input parameter{in}	
trig_type	trigger type, refer to <a href="#">Table 3-403. Enum exti_trig_type_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

## exti\_interrupt\_enable

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

**Table 3-406. 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 3-401. Enum exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_interrupt_enable(EXTI_0);
```

### exti\_interrupt\_disable

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

**Table 3-407. Function exti\_interrupt\_disable**

Function name	exti_interrupt_disable
Function prototype	void exti_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the interrupts from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-401. Enum exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_interrupt_disable(EXTI_0);
```

### exti\_event\_enable

The description of exti\_event\_enable is shown as below:

**Table 3-408. Function exti\_event\_enable**

Function name	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 3-401. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### exti\_event\_disable

The description of exti\_event\_disable is shown as below:

**Table 3-409. Function exti\_event\_disable**

<b>Function name</b>	exti_event_disable
<b>Function prototype</b>	void exti_event_disable(exti_line_enum linex);
<b>Function descriptions</b>	disable the events from EXTI line x
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>linex</b>	EXTI line x, refer to <a href="#">Table 3-401. Enum exti_line_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### exti\_software\_interrupt\_enable

The description of exti\_software\_interrupt\_enable is shown as below:

**Table 3-410. Function exti\_software\_interrupt\_enable**

<b>Function name</b>	exti_software_interrupt_enable
<b>Function prototype</b>	void exti_software_interrupt_enable(exti_line_enum linex);
<b>Function descriptions</b>	enable the software interrupt event from EXTI line x

Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-401. Enum 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 3-411. Function exti\_software\_interrupt\_disable**

Function name	exti_software_interrupt_disable
Function prototype	void exti_software_interrupt_disable(exti_line_enum linex);
Function descriptions	disable the software interrupt event from EXTI line x
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-401. Enum exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### exti\_flag\_get

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

**Table 3-412. Function exti\_flag\_get**

Function name	exti_flag_get
Function prototype	FlagStatus exti_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-

Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-401. Enum exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

### exti\_flag\_clear

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

**Table 3-413. Function exti\_flag\_clear**

Function name	exti_flag_clear
Function prototype	void exti_flag_clear(exti_line_enum linex);
Function descriptions	clear EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-401. Enum exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### exti\_interrupt\_flag\_get

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

**Table 3-414. Function exti\_interrupt\_flag\_get**

Function name	exti_interrupt_flag_get
Function prototype	FlagStatus exti_interrupt_flag_get(exti_line_enum linex);
Function descriptions	get EXTI line x interrupt pending flag
Precondition	-
The called functions	-
Input parameter{in}	
linex	EXTI line x, refer to <a href="#">Table 3-401. Enum exti_line_enum</a>

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

Example:

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

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

### exti\_interrupt\_flag\_clear

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

**Table 3-415. 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 3-401. Enum exti_line_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
exti_interrupt_flag_clear(EXTI_0);
```

## 3.15. FAC

The filter arithmetic accelerator unit can realize finite impulse response (FIR) filters and infinite impulse response (IIR) filters. The FAC registers are listed in chapter [3.15.1](#), the FAC firmware functions are introduced in chapter [3.15.2](#).

### 3.15.1. Descriptions of Peripheral registers

FAC registers are listed in the table shown as below:

**Table 3-416. FAC Registers**

Registers	Descriptions
FAC_X0BCFG	FAC X0 buffer configure register



Registers	Descriptions
FAC_X1BCFG	FAC X1 buffer configure register
FAC_YBCFG	FAC Y buffer configure register
FAC_PARACFG	FAC Parameter configure register
FAC_CTL	FAC control register
FAC_STAT	FAC status register
FAC_WDATA	FAC write data register
FAC_RDATA	FAC read data register

### 3.15.2. Descriptions of Peripheral functions

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

**Table 3-417. FAC firmware function**

Function name	Function description
fac_deinit	reset the FAC peripheral
fac_struct_para_init	initialize the FAC filter parameter struct with the default values
fac_fixed_data_preload_init	initialize the FAC fixed data preload parameter struct with the default values
fac_float_data_preload_init	initialize the FAC float data preload parameter struct with the default values
fac_init	initialize the FAC peripheral
fac_fixed_buffer_preload	FAC preload X0 X1 Y fixed buffer
fac_float_buffer_preload	FAC preload X0 X1 Y float buffer
fac_fixed_data_preload	FAC preload fixed data
fac_float_data_preload	FAC preload float data
fac_reset	FAC reset write and read pointers
fac_clip_config	config the FAC clip feature
fac_float_enable	enable FAC float point format
fac_float_disable	disable FAC float point format
fac_dma_enable	enable the FAC dma
fac_dma_disable	disable the FAC dma
fac_x0_config	FAC config input buffer
fac_x1_config	FAC config coefficient buffer
fac_y_config	FAC config output buffer
fac_function_config	FAC config function execute
fac_start	start the FAC
fac_stop	stop the FAC
fac_finish_calculate	finish the filter calculate
fac_interrupt_enable	enable the FAC interrupt

Function name	Function description
fac_interrupt_disable	disable the FAC interrupt
fac_interrupt_flag_get	get the FAC interrupt flag status
fac_flag_get	get the FAC flag status
fac_fixed_data_write	FAC write data with fixed point format
fac_fixed_data_read	FAC read data with fixed point format
fac_float_data_write	FAC write data with float point format
fac_float_data_read	FAC read data with fixed point format

### Structure fac\_parameter\_struct

**Table 3-418. Structure fac\_parameter\_struct**

Member name	Function description
input_addr	base address of the input buffer (X0)
input_size	size of input buffer
coeff_addr	base address of the coefficient buffer (X1)
coeff_size	size of coefficient buffer
output_addr	base address of the output buffer (Y)
output_size	size of output buffer
ipp	vector IPP length
ipq	vector IPQ length
ipr	vector IPR length
input_threshold	threshold of input buffer full
output_threshold	threshold of output buffer empty
clip	enable or disable the clipping feature
func	FAC functions select

### Structure fac\_fixed\_data\_preload\_struct

**Table 3-419. Structure fac\_fixed\_data\_preload\_struct**

Member name	Function description
coeffa_size	size of the coefficient vector A
*coeffa_ctx	context of the coefficient vector A(int16_t format)
coeffb_size	size of the coefficient vector B
*coeffb_ctx	context of the coefficient vector B(int16_t format)
input_size	size of the input data
*input_ctx	context of the input data(int16_t format)
output_size	size of the output data
*output_ctx	context of the output data(int16_t format)

## Structure fac\_float\_data\_preload\_struct

**Table 3-420. Structure fac\_float\_data\_preload\_struct**

Member name	Function description
coeffa_size	size of the coefficient vector A
*coeffa_ctx	context of the coefficient vector A(float format)
coeffb_size	size of the coefficient vector B
*coeffb_ctx	context of the coefficient vector B(float format)
input_size	size of the input data
*input_ctx	context of the input data(float format)
output_size	size of the output data
*output_ctx	context of the output data(float format)

## fac\_deinit

The description of fac\_deinit is shown as below:

**Table 3-421. Function fac\_deinit**

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

Example:

```
/* deinitialize FAC */
```

```
fac_deinit();
```

## fac\_struct\_para\_init

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

**Table 3-422. Function fac\_struct\_para\_init**

Function name	fac_struct_para_init
Function prototype	void fac_struct_para_init(fac_parameter_struct* fac_parameter);
Function descriptions	initialize the FAC filter parameter struct with the default values
Precondition	-
The called functions	-
Input parameter{in}	

<b>fac_parameter</b>	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_parameter_struct facconfig;
```

```
fac_struct_para_init(&facconfig);
```

### fac\_fixed\_data\_preload\_init

The description of fac\_fixed\_data\_preload\_init is shown as below:

**Table 3-423. Function fac\_fixed\_data\_preload\_init**

<b>Function name</b>	fac_fixed_data_preload_init
<b>Function prototype</b>	void fac_fixed_data_preload_init(fac_fixed_data_preload_struct *init_struct);
<b>Function descriptions</b>	initialize the FAC fixed data preload parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>init_struct</b>	FAC fixed data preload init parameter struct, the structure members can refer to <a href="#">Structure fac_fixed_data_preload_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_fixed_data_preload_struct init_struct;
```

```
fac_fixed_data_preload_init (&init_struct);
```

### fac\_float\_data\_preload\_init

The description of fac\_float\_data\_preload\_init is shown as below:

**Table 3-424. Function fac\_float\_data\_preload\_init**

<b>Function name</b>	fac_float_data_preload_init
<b>Function prototype</b>	void fac_float_data_preload_init(fac_float_data_preload_struct *init_struct);
<b>Function descriptions</b>	initialize the FAC float data preload parameter struct with the default values

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>init_struct</b>	FAC float data preload init parameter struct, the structure members can refer to <a href="#">Structure fac_float_data_preload_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize FAC init parameter struct with a default value */
```

```
fac_float_data_preload_struct init_struct;
```

```
fac_float_data_preload_init (&init_struct);
```

## fac\_init

The description of fac\_init is shown as below:

**Table 3-425. Function fac\_init**

<b>Function name</b>	fac_init
<b>Function prototype</b>	void fac_init(fac_parameter_struct* fac_parameter);
<b>Function descriptions</b>	Initialize the FAC peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fac_parameter</b>	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* Initialize the FAC peripheral */
```

```
fac_parameter_struct facconfig;
```

```
fac_init(&facconfig);
```

## fac\_fixed\_buffer\_preload

The description of fac\_fixed\_buffer\_preload is shown as below:

Table 3-426. Function fac\_preload

Function name	fac_fixed_buffer_preload
Function prototype	void fac_fixed_buffer_preload(fac_fixed_data_preload_struct* init_struct);
Function descriptions	FAC preload X0 X1 Y fixed buffer
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_fixed_data_preload_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC preload X0 X1 Y fixed buffer */
fac_fixed_data_preload_struct faccoeff;
fac_fixed_buffer_preload (&faccoeff);
```

### fac\_float\_buffer\_preload

The description of fac\_float\_buffer\_preload is shown as below:

Table 3-427. Function fac\_float\_buffer\_preload

Function name	fac_float_buffer_preload
Function prototype	void fac_float_buffer_preload(fac_float_data_preload_struct* init_struct);
Function descriptions	FAC preload X0 X1 Y float buffer
Precondition	-
The called functions	-
Input parameter{in}	
init_struct	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_float_data_preload_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* FAC preload X0 X1 Y float buffer */
fac_float_data_preload_struct faccoeff;
fac_float_buffer_preload(&faccoeff);
```

## fac\_fixed\_data\_preload

The description of fac\_fixed\_data\_preload is shown as below:

**Table 3-428. Function fac\_float\_preload**

<b>Function name</b>	fac_fixed_data_preload
<b>Function prototype</b>	void fac_fixed_data_preload(uint8_t size, int16_t *data);
<b>Function descriptions</b>	FAC preload fixed data pointer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>size</b>	size of data
<b>Input parameter{in}</b>	
<b>*data</b>	16-bit data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* FAC preload context of the coefficient vector B */
fac_fixed_data_preload_struct init_struct;
fac_fixed_data_preload(init_struct->coeffB_size,(init_struct->coeffB_ctx));
```

## fac\_float\_data\_preload

The description of fac\_float\_data\_preload is shown as below:

**Table 3-429. Function fac\_float\_preload**

<b>Function name</b>	fac_float_data_preload
<b>Function prototype</b>	void fac_float_data_preload(uint8_t size, float *data);
<b>Function descriptions</b>	FAC preload float data pointer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>size</b>	size of data
<b>Input parameter{in}</b>	
<b>*data</b>	32-bit data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* FAC preload context of the coefficient vector B */
```

```
fac_float_data_preload_struct init_struct;
```

```
fac_float_data_preload (init_struct->coeffB_size,(init_struct->coeffB_ctx));
```

## fac\_reset

The description of fac\_reset is shown as below:

**Table 3-430. Function fac\_init**

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

Example:

```
/* FAC reset write and read pointers */
```

```
fac_reset();
```

## fac\_clip\_config

The description of fac\_clip\_config is shown as below:

**Table 3-431. Function fac\_clip\_config**

<b>Function name</b>	fac_clip_config
<b>Function prototype</b>	void fac_clip_config(uint8_t cpmode);
<b>Function descriptions</b>	config the FAC clip feature
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cpmode</b>	the state of clip
<i>FAC_CP_ENABLED</i>	enable clip
<i>FAC_CP_DISABLE</i>	disable clip
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-



Example:

```
/* config the FAC clip enable */
fac_clip_config(FAC_CP_ENABLE);
```

### fac\_float\_enable

The description of fac\_float\_enable is shown as below:

**Table 3-432. Function fac\_init**

<b>Function name</b>	fac_float_enable
<b>Function prototype</b>	void fac_float_enable(void);
<b>Function descriptions</b>	enable FAC float point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FAC float point format */
fac_float_enable ();
```

### fac\_float\_disable

The description of fac\_float\_disable is shown as below:

**Table 3-433. Function fac\_float\_disable**

<b>Function name</b>	fac_float_disable
<b>Function prototype</b>	void fac_float_disable(void);
<b>Function descriptions</b>	disable FAC float point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FAC float point format */
```

```
fac_float_disable ();
```

## fac\_dma\_enable

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

**Table 3-434. Function fac\_dma\_enable**

<b>Function name</b>	fac_dma_enable
<b>Function prototype</b>	void fac_dma_enable(uint32_t dma_req);
<b>Function descriptions</b>	enable the FAC DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_req</b>	transfer type
FAC_DMA_READ	enable DMA read buffer
FAC_DMA_WRITE	enable DMA write buffer
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the FAC read buffer by DMA */
fac_dma_enable(FAC_DMA_READ);
```

## fac\_dma\_disable

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

**Table 3-435. Function fac\_dma\_disable**

<b>Function name</b>	fac_dma_disable
<b>Function prototype</b>	void fac_dma_disable(uint32_t dma_req);
<b>Function descriptions</b>	disable the FAC DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dma_req</b>	transfer type
FAC_DMA_READ	disable DMA read buffer
FAC_DMA_WRITE	disable DMA write buffer
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the FAC read buffer by DMA */
```

```
fac_dma_disable(FAC_DMA_READ);
```

### fac\_x0\_config

The description of fac\_x0\_config is shown as below:

**Table 3-436. Function fac\_dma\_enable**

<b>Function name</b>	fac_x0_config
<b>Function prototype</b>	void fac_x0_config(uint32_t watermark, uint8_t baseaddr, uint8_t bufsize);
<b>Function descriptions</b>	FAC config input buffer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>watermark</b>	threshold of input buffer
<i>FAC_THRESHOLD_x</i>	X0 buffer full threshold(x =1,2,4,8)
<b>Input parameter{in}</b>	
<b>baseaddr</b>	base address of input buffer, 0..255
<b>Input parameter{in}</b>	
<b>bufsize</b>	buffer size of input buffer, 0..255
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config input buffer threshold of watermark, base address, buffer size */
```

```
fac_x0_config(FAC_THRESHOLD_1,20,10);
```

### fac\_x1\_config

The description of fac\_x1\_config is shown as below:

**Table 3-437. Function fac\_dma\_enable**

<b>Function name</b>	fac_x1_config
<b>Function prototype</b>	void fac_x1_config(uint8_t baseaddr, uint8_t bufsize);
<b>Function descriptions</b>	FAC config coefficient buffer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>baseaddr</b>	base address of coefficientbuffer, 0..255
<b>Input parameter{in}</b>	
<b>bufsize</b>	buffer size of coefficient buffer, 0..255
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* config coefficient buffer base address and buffer size */
```

```
fac_x1_config(10,10);
```

### fac\_y\_config

The description of fac\_y\_config is shown as below:

**Table 3-438. Function fac\_dma\_enable**

<b>Function name</b>	fac_y_config
<b>Function prototype</b>	void fac_y_config(uint32_t watermark, uint8_t baseaddr, uint8_t bufsize);
<b>Function descriptions</b>	FAC config output buffer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>watermark</b>	threshold of output buffer
<i>FAC_THRESHOLD_x</i>	Y buffer empty threshold(x =1,2,4,8)
<b>Input parameter{in}</b>	
<b>baseaddr</b>	base address of outputbuffer, 0..255
<b>Input parameter{in}</b>	
<b>bufsize</b>	buffer size of output buffer, 0..255
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config output buffer threshold of watermark, base address, buffer size */
```

```
fac_y_config(FAC_THRESHOLD_1,30,20);
```

### fac\_function\_config

The description of fac\_function\_config is shown as below:

**Table 3-439. Function fac\_dma\_enable**

<b>Function name</b>	fac_function_config
<b>Function prototype</b>	void fac_function_config(fac_parameter_struct* fac_parameter);
<b>Function descriptions</b>	FAC config function execute
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
fac_parameter	FAC init parameter struct, the structure members can refer to <a href="#">Structure fac_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* config FAC work in FIR mode*/

fac_parameter_struct facconfig;

facconfig.func = FUNC_CONVO_FIR;

facconfig.ipp = fir_coeffb_size;

facconfig.ipq = 0;

facconfig.ipr = fir_gain;

fac_function_config(&facconfig);

```

## fac\_start

The description of fac\_start is shown as below:

**Table 3-440. Function fac\_start**

Function name	fac_start
Function prototype	void fac_start(void);
Function descriptions	start the FAC
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* start the FAC*/

fac_start();

```

## fac\_stop

The description of fac\_stop is shown as below:

**Table 3-441. Function fac\_start**

Function name	fac_stop
Function prototype	void fac_stop(void);
Function descriptions	stop the FAC
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* stop the FAC*/
```

```
fac_stop();
```

### fac\_finish\_calculate

The description of fac\_finish\_calculate is shown as below:

**Table 3-442. Function fac\_start**

Function name	fac_finish_calculate
Function prototype	void fac_finish_calculate(void);
Function descriptions	finish the filter calculate
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* finish the filter calculate*/
```

```
fac_finish_calculate ();
```

### fac\_fixed\_data\_write

The description of fac\_fixed\_data\_write is shown as below:

**Table 3-443. Function fac\_fixed\_data\_write**

Function name	fac_fixed_data_write
---------------	----------------------

<b>Function prototype</b>	void fac_fixed_data_write(int16_t data);
<b>Function descriptions</b>	FAC write data with fixed point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data</b>	16-bit data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* FAC fixed data write */
fac_fixed_data_write(300);
```

### fac\_fixed\_data\_read

The description of fac\_float\_data\_read is shown as below:

**Table 3-444. Function fac\_fixed\_data\_write**

<b>Function name</b>	fac_fixed_data_read
<b>Function prototype</b>	int16_t fac_fixed_data_read(void);
<b>Function descriptions</b>	FAC read data with fixed point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int16_t</b>	FAC int16_t data value

Example:

```
/* FAC fixed data read */
int16_t data;
data = fac_fixed_data_read();
```

### fac\_float\_data\_write

The description of fac\_float\_data\_write is shown as below:

**Table 3-445. Function fac\_float\_data\_write**

<b>Function name</b>	fac_float_data_write
<b>Function prototype</b>	void fac_float_data_write(float data);

<b>Function descriptions</b>	FAC write data with float ponit format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data</b>	32-bit data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* FAC float data write */
fac_float_data_write(200.0f);
```

### fac\_float\_data\_read

The description of fac\_float\_data\_read is shown as below:

**Table 3-446. Function fac\_float\_data\_read**

<b>Function name</b>	fac_float_data_read
<b>Function prototype</b>	int16_t fac_float_data_read(void);
<b>Function descriptions</b>	FAC read data with float point format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>float</b>	FAC float data value

Example:

```
/* FAC float data read */
float data;
data = fac_float_data_read();
```

### fac\_interrupt\_enable

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

**Table 3-447. Function fac\_interrupt\_enable**

<b>Function name</b>	fac_interrupt_enable
<b>Function prototype</b>	void fac_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable the FAC interrupt



<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FAC interrupt
<i>FAC_CTL_RIE</i>	read buffer interrupt
<i>FAC_CTL_WIE</i>	write buffer interrupt
<i>FAC_CTL_OFEIE</i>	overflow error interrupt
<i>FAC_CTL_UFEIE</i>	underflow error interrupt
<i>FAC_CTL_STEIE</i>	saturation error interrupt
<i>FAC_CTL_GSTEIE</i>	gain saturation error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the FAC read buffer interrupt*/
```

```
fac_interrupt_enable(FAC_CTL_RIE);
```

### fac\_interrupt\_disable

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

**Table 3-448. Function fac\_interrupt\_disable**

<b>Function name</b>	fac_interrupt_disable
<b>Function prototype</b>	void fac_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable the FAC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FAC interrupt
<i>FAC_CTL_RIE</i>	read buffer interrupt
<i>FAC_CTL_WIE</i>	write buffer interrupt
<i>FAC_CTL_OFEIE</i>	overflow error interrupt
<i>FAC_CTL_UFEIE</i>	underflow error interrupt
<i>FAC_CTL_STEIE</i>	saturation error interrupt
<i>FAC_CTL_GSTEIE</i>	gain saturation error interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the FAC read buffer interrupt*/
```

```
fac_interrupt_disable(FAC_CTL_RIE);
```

### fac\_interrupt\_flag\_get

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

**Table 3-449. Function fac\_interrupt\_flag\_get**

<b>Function name</b>	fac_interrupt_flag_get
<b>Function prototype</b>	FlagStatus fac_interrupt_flag_get(uint8_t interrupt);
<b>Function descriptions</b>	get FAC interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FAC interrupt flag status
<i>FAC_INT_FLAG_YBEF</i>	Y buffer empty interrupt flag
<i>FAC_INT_FLAG_X0BF</i> <i>F</i>	X0 buffer full interrupt flag
<i>FAC_INT_FLAG_OFEF</i>	overflow error interrupt flag
<i>FAC_INT_FLAG_UFEF</i>	underflow error interrupt flag
<i>FAC_INT_FLAG_STEF</i>	saturation error interrupt flag
<i>FAC_INT_FLAG_GSTE</i> <i>F</i>	gain saturation error interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get Y buffer empty flag status */
```

```
fac_interrupt_flag_get(FAC_INT_FLAG_YBEF);
```

### fac\_flag\_get

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

**Table 3-450. Function fac\_flag\_get**

<b>Function name</b>	fac_flag_get
<b>Function prototype</b>	FlagStatus fac_flag_get(uint32_t flag);
<b>Function descriptions</b>	get FAC flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FAC interrupt flag status
<i>FAC_FLAG_YBEF</i>	Y buffer empty flag
<i>FAC_FLAG_X0BFF</i>	X0 buffer full flag

<i>FAC_FLAG_OFEF</i>	overflow error flag
<i>FAC_FLAG_UFEF</i>	underflow error flag
<i>FAC_FLAG_STEF</i>	saturation error flag
<i>FAC_FLAG_GSTEF</i>	gain saturation error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get Y buffer empty flag status */
fac_flag_get(FAC_FLAG_YBEF);
```

## 3.16. FMC

There is flash controller and option byte. The FMC registers are listed in chapter [3.16.1](#), the FMC firmware functions are introduced in chapter [3.16.2](#).

### 3.16.1. Descriptions of Peripheral registers

FMC registers are listed in the table shown as below:

**Table 3-451. FMC Registers**

Registers	Descriptions
FMC_KEY	FMC unlock key register
FMC_OBKEY	FMC option byte unlock key register
FMC_CTL	FMC control register
FMC_STAT	FMC status register
FMC_ADDR	FMC address register
FMC_OBCTL	FMC option byte control register
FMC_OBSTAT0_EFT	FMC effective option byte status0 register
FMC_OBSTAT0_MDF	FMC modified option byte status0 register
FMC_DCRPADDR_EFT	FMC effective DCRP address register
FMC_DCRPADDR_MDF	FMC modified DCRP address register
FMC_SCRADDR_EFT	FMC effective secure address register
FMC_SCRADDR_MDF	FMC modified secure address register
FMC_WP_EFT	FMC effective erase/program protection register

Registers	Descriptions
FMC_WP_MDF	FMC modified erase/program protection register
FMC_BTADDR_EFT	FMC effective boot address register
FMC_BTADDR_MDF	FMC modified boot address register
FMC_OBSTAT1_EFT	FMC effective option byte status1 register
FMC_OBSTAT1_MDF	FMC modified option byte status1 register
FMC_NODEC	FMC NO-RTDEC area register
FMC_ECCADDR	FMC ECC error address register
FMC_AESIVx_EFT(x = 0...2)	FMC effective AES IVx register(x = 0...2)
FMC_AESIVx_MDF(x = 0...2)	FMC modified AES IVx register(x = 0...2)
EFUSE_PIDx(x = 0,1)	FMC product ID register x(x = 0,1)

### 3.16.2. Descriptions of Peripheral functions

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

**Table 3-452. FMC firmware function**

Function name	Function description
fmc_unlock	unlock FMC_CTL register
fmc_lock	lock FMC_CTL register
fmc_sector_erase	FMC erase sector
fmc_typical_mass_erase	FMC typical mass erase
fmc_protection_removed_mass_erase	FMC protection-removed mass erase
fmc_word_program	FMC program a word at the corresponding address
fmc_doubleword_program	FMC program a double-word at the corresponding address
fmc_check_programming_area_enable	enable check programming area
fmc_check_programming_area_disable	disable check programming area
ob_unlock	unlock the option byte operation
ob_lock	lock the option byte operation
ob_start	send option bytes modification start command
ob_factory_value_config	modify option byte to factory value
ob_secure_access_mode_enable	enable secure access mode
ob_secure_access_mode_disable	disable secure access mode
ob_security_protection_config	configure the option byte security protection level
ob_bor_threshold_config	configure option byte BOR threshold value

Function name	Function description
ob_low_power_config	configure low power related option byte
ob_tcm_ecc_config	configure TCM ECC option byte
ob_iospeed_optimize_config	configure I/O speed optimization option byte
ob_tcm_shared_ram_config	configure option byte TCM shared RAM size
ob_data_program	modify option byte DATA
ob_boot_address_config	configure boot address
ob_dcrp_area_config	configure DCRP area
ob_secure_area_config	configure secure-access area
ob_write_protection_enable	enable erase/program protection
ob_write_protection_disable	disable erase/program protection
ob_secure_mode_get	get the option byte secure access mode
ob_security_protection_flag_get	get the option byte security protection level
ob_bor_threshold_get	get the option byte BOR threshold value
ob_low_power_get	get low power related option byte
ob_tcm_ecc_get	get TCM ECC configuration
ob_iospeed_optimize_get	get IO speed optimize configuration
ob_itcm_shared_ram_size_get	get the option byte ITCM shared RAM size
ob_dtcn_shared_ram_size_get	get the option byte DTCM shared RAM size
ob_data_get	get user data value
ob_boot_address_get	get boot address
ob_dcrp_area_get	get DCRP area configuration
ob_secure_area_get	get secure-access area configuration
ob_write_protection_get	get the option byte erase/program protection configuration
fmc_no_rtdec_config	configure NO-RTDEC area
fmc_aes_iv_config	configure aes initialization vector
fmc_flash_ecc_get	get Flash ECC function enable flag
fmc_no_rtdec_get	get NO-RTDEC area
fmc_aes_iv_get	get AES initialization vector
fmc_pid_get	get product ID
fmc_flag_get	get FMC flag status
fmc_flag_clear	clear FMC flag
fmc_interrupt_enable	enable FMC interrupt
fmc_interrupt_disable	disable FMC interrupt
fmc_interrupt_flag_get	get FMC interrupt flag status
fmc_interrupt_flag_clear	clear FMC interrupt flag status

## Enum fmc\_state\_enum

Table 3-453. Enum fmc\_state\_enum

enum name	enum description
FMC_READY	the operation has been completed
FMC_BUSY	the operation is in progress

enum name	enum description
FMC_WPERR	erase/program protection error
FMC_PGSE	program sequence error
FMC_RPERR	read protection error
FMC_RSERR	read secure error
FMC_ECCCOR	one bit correct error
FMC_ECCDET	two bits detect error
FMC_OBMERR	option byte modify error
FMC_TOERR	timeout error

### Enum fmc\_flag\_enum

Table 3-454. Enum fmc\_flag\_enum

enum name	enum description
FMC_FLAG_BUSY	flash busy flag
FMC_FLAG_END	flash end of operation flag
FMC_FLAG_WPER R	flash erase/program protection error flag
FMC_FLAG_PGSE RR	flash program sequence error flag
FMC_FLAG_RPER R	flash read protection error flag
FMC_FLAG_RSER R	flash read secure error flag
FMC_FLAG_ECCC OR	flash one bit correct error flag
FMC_FLAG_ECCD ET	flash two bits detect error flag
FMC_FLAG_OBME RR	option byte modify error flag
FMC_FLAG_FECC	flash ECC function flag

### Enum fmc\_interrupt\_flag\_enum

Table 3-455. Enum fmc\_interrupt\_flag\_enum

enum name	enum description
FMC_INT_FLAG_E ND	flash end of operation interrupt flag
FMC_INT_FLAG_W PERR	flash erase/program protection error interrupt flag
FMC_INT_FLAG_P GSERR	flash program sequence error interrupt flag
FMC_INT_FLAG_R PERR	flash read protection error interrupt flag

enum name	enum description
FMC_INT_FLAG_R SERR	flash read secure error interrupt flag
FMC_INT_FLAG_E CCCOR	flash one bit error detected and correct interrupt flag
FMC_INT_FLAG_E CCDET	flash two bit errors detect interrupt flag
FMC_INT_FLAG_O BMERR	option byte modify error flag

### Enum fmc\_interrupt\_enum

**Table 3-456. Enum fmc\_interrupt\_enum**

enum name	enum description
FMC_INT_END	FMC end of operation flag
FMC_INT_WPERR	FMC erase/program protection error flag
FMC_INT_PGSERR	FMC program sequence error flag
FMC_INT_RPERR	FMC read protection error flag
FMC_INT_RSERR	FMC read secure error flag
FMC_INT_ECCCO R	FMC one bit correct error flag
FMC_INT_ECCDET	FMC two bits detect error flag
FMC_INT_OBMER R	FMC option byte modify error flag

### fmc\_unlock

The description of fmc\_unlock is shown as below:

**Table 3-457. Function fmc\_unlock**

<b>Function name</b>	fmc_unlock
<b>Function prototype</b>	void fmc_unlock(void);
<b>Function descriptions</b>	unlock FMC_CTL register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* unlock FMC_CTL register */
```

```
fmc_unlock();
```

## fmc\_lock

The description of fmc\_lock is shown as below:

**Table 3-458. Function fmc\_lock**

<b>Function name</b>	fmc_lock
<b>Function prototype</b>	void fmc_lock(void);
<b>Function descriptions</b>	lock FMC_CTL register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* lock FMC_CTL register */
```

```
fmc_lock();
```

## fmc\_sector\_erase

The description of fmc\_sector\_erase is shown as below:

**Table 3-459. Function fmc\_page\_erase**

<b>Function name</b>	fmc_sector_erase
<b>Function prototype</b>	fmc_state_enum fmc_sector_erase(uint32_t address);
<b>Function descriptions</b>	FMC erase sector
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>address</b>	address to erase
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();
```

```
/* erase sector */
```

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



## fmc\_typical\_mass\_erase

The description of fmc\_typical\_mass\_erase is shown as below:

**Table 3-460. Function fmc\_mass\_erase**

<b>Function name</b>	fmc_typical_mass_erase
<b>Function prototype</b>	fmc_state_enum fmc_typical_mass_erase(void);
<b>Function descriptions</b>	FMC typical mass erase
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();

/* FMC typical mass erase */

fmc_state_enum state = fmc_typical_mass_erase();
```

## fmc\_protection\_removed\_mass\_erase

The description of fmc\_protection\_removed\_mass\_erase is shown as below:

**Table 3-461. Function fmc\_protection\_removed\_mass\_erase**

<b>Function name</b>	fmc_protection_removed_mass_erase
<b>Function prototype</b>	fmc_state_enum fmc_protection_removed_mass_erase(void);
<b>Function descriptions</b>	FMC protection-removed mass erase
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();

/* FMC protection-removed mass erase */

fmc_state_enum state = fmc_protection_removed_mass_erase();
```

## fmc\_word\_program

The description of fmc\_word\_program is shown as below:

**Table 3-462. Function fmc\_word\_program**

<b>Function name</b>	fmc_word_program
<b>Function prototype</b>	fmc_state_enum fmc_word_program(uint32_t address, uint32_t data);
<b>Function descriptions</b>	FMC program a word at the corresponding address
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>address</b>	address to program
<b>Input parameter{in}</b>	
<b>data</b>	word to program
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();
```

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

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

```
fmc_state_enum fmc_state = fmc_doubleword_program(0x08004000, 0x11223344);
```

## fmc\_doubleword\_program

The description of fmc\_doubleword\_program is shown as below:

**Table 3-463. Function fmc\_doubleword\_program**

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

<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>
-----------------------	---

Example:

```
fmc_unlock();
```

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

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

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

### fmc\_check\_programming\_area\_enable

The description of fmc\_check\_programming\_area\_enable is shown as below:

**Table 3-464. Function fmc\_check\_programming\_area\_enable**

<b>Function name</b>	fmc_check_programming_area_enable
<b>Function prototype</b>	fmc_state_enum fmc_check_programming_area_enable(void);
<b>Function descriptions</b>	enable check programming area
<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, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();
```

```
/* enable check programming area before program operation */
```

```
fmc_state_enum fmc_state = fmc_check_programming_area_enable();
```

### fmc\_check\_programming\_area\_disable

The description of fmc\_check\_programming\_area\_disable is shown as below:

**Table 3-465. Function fmc\_check\_programming\_area\_disable**

<b>Function name</b>	fmc_check_programming_area_disable
<b>Function prototype</b>	fmc_state_enum fmc_check_programming_area_disable(void);
<b>Function descriptions</b>	disable check programming area
<b>Precondition</b>	fmc_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-

Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();
```

```
/* disable check programming area before program operation */
```

```
fmc_state_enum fmc_state = fmc_check_programming_area_disable();
```

## ob\_unlock

The description of ob\_unlock is shown as below:

**Table 3-466. Function ob\_unlock**

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

Example:

```
fmc_unlock();
```

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

```
ob_unlock();
```

## ob\_lock

The description of ob\_lock is shown as below:

**Table 3-467. Function ob\_lock**

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

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

Example:

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

```
ob_lock();
```

### ob\_start

The description of ob\_start is shown as below:

**Table 3-468. Function ob\_start**

<b>Function name</b>	ob_start
<b>Function prototype</b>	fmc_state_enum ob_start(void);
<b>Function descriptions</b>	send option bytes modification start command
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* start modify WP option bytes */
```

```
fmc_state = ob_write_protection_disable(OB_WP_1);
```

```
ob_start();
```

### ob\_factory\_value\_config

The description of ob\_factory\_value\_config is shown as below:

**Table 3-469. Function ob\_factory\_value\_config**

<b>Function name</b>	ob_factory_value_config
<b>Function prototype</b>	fmc_state_enum ob_factory_value_config(void);

<b>Function descriptions</b>	modify option byte to factory value
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	ob_start
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* modify option byte to default value */

fmc_state = ob_default_init();
```

### ob\_secure\_access\_mode\_enable

The description of ob\_secure\_access\_mode\_enable is shown as below:

**Table 3-470. Function ob\_secure\_access\_mode\_enable**

<b>Function name</b>	ob_secure_access_mode_enable
<b>Function prototype</b>	fmc_state_enum ob_secure_access_mode_enable(void);
<b>Function descriptions</b>	enable secure access mode
<b>Precondition</b>	ob_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, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* enable secure access mode */

fmc_state = ob_secure_access_mode_enable();
```

## ob\_secure\_access\_mode\_disable

The description of ob\_secure\_access\_mode\_disable is shown as below:

**Table 3-471. Function ob\_secure\_access\_mode\_disable**

<b>Function name</b>	ob_secure_access_mode_disable
<b>Function prototype</b>	fmc_state_enum ob_secure_access_mode_disable(void);
<b>Function descriptions</b>	disable secure access mode
<b>Precondition</b>	ob_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, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* disable security protection */

fmc_state = ob_secure_access_mode_disable();
```

## ob\_security\_protection\_config

The description of ob\_security\_protection\_config is shown as below:

**Table 3-472. 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 the option byte security protection level
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_spc</b>	specify security protection level
<i>FMC_NSPC</i>	no protection
<i>FMC_LSPC</i>	protection level low
<i>FMC_HSPC</i>	protection level high
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* enable security protection */

fmc_state = ob_security_protection_config(FMC_LSPC);
```

### ob\_bor\_threshold\_config

The description of ob\_bor\_threshold\_config is shown as below:

**Table 3-473. Function ob\_bor\_threshold\_config**

<b>Function name</b>	ob_bor_threshold_config
<b>Function prototype</b>	fmc_state_enum ob_bor_threshold_config(uint32_t ob_bor_th);
<b>Function descriptions</b>	configure the option byte BOR threshold value
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_bor_th</b>	option byte BOR threshold value
OB_BOR_TH_VALUE3	BOR threshold value 3
OB_BOR_TH_VALUE2	BOR threshold value 2
OB_BOR_TH_VALUE1	BOR threshold value 1
OB_BOR_TH_OFF	no BOR function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* set BOR threshold value 3 */

fmc_state = ob_bor_threshold_config(OB_BOR_TH_VALUE3);
```

### ob\_low\_power\_config

The description of ob\_low\_power\_config is shown as below:

**Table 3-474. Function ob\_low\_power\_config**

<b>Function name</b>	ob_low_power_config
----------------------	---------------------



<b>Function prototype</b>	fmc_state_enum ob_low_power_config(uint32_t ob_fwdgt, uint32_t ob_deepsleep, uint32_t ob_stdby, uint32_t ob_fwdg_suspend_deepsleep, uint32_t ob_fwdg_suspend_standby);
<b>Function descriptions</b>	configure low power related option byte
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_fwdgt</b>	option byte watchdog value
<i>OB_FWDGT_SW</i>	software free watchdog
<i>OB_FWDGT_HW</i>	hardware free watchdog
<b>Input parameter{in}</b>	
<b>ob_deepsleep</b>	option byte deepsleep reset value
<i>OB_DEEPSLEEP_NRS</i> <i>T</i>	no reset when entering deepsleep mode
<i>OB_DEEPSLEEP_RST</i>	generate a reset instead of entering deepsleep mode
<b>Input parameter{in}</b>	
<b>ob_stdby</b>	option byte standby reset value
<i>OB_STDBY_NRST</i>	no reset when entering standby mode
<i>OB_STDBY_RST</i>	generate a reset instead of entering standby mode
<b>Input parameter{in}</b>	
<b>ob_fwdg_suspend_deepsleep</b>	option byte FWDG suspend status in deep-sleep mode
<i>OB_DPSLP_FWDGT_SUSPEND</i>	free watchdog is suspended in deep-sleep mode
<i>OB_DPSLP_FWDGT_RUN</i>	free watchdog is running in deep-sleep mode
<b>Input parameter{in}</b>	
<b>ob_fwdg_suspend_standby</b>	option byte FWDG suspend status in standby mode
<i>OB_STDBY_FWDGT_SUSPEND</i>	free watchdog is suspended in standby mode
<i>OB_STDBY_FWDGT_RUN</i>	free watchdog is running in standby mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();
```

```
/* configure low power related option byte */
```

```
fmc_state = fmc_state_enum ob_low_power_config(OB_FWDGT_SW, OB_DEEPSLEEP
_NRST, OB_STDBY_NRST, OB_DPSLP_FWDGT_SUSPEND, OB_STDBY_FWDGT_SU
SPEND);
```

### ob\_tcm\_ecc\_config

The description of ob\_tcm\_ecc\_config is shown as below:

**Table 3-475. Function ob\_tcm\_ecc\_config**

<b>Function name</b>	ob_tcm_ecc_config
<b>Function prototype</b>	fmc_state_enum ob_tcm_ecc_config(uint32_t ob_itcmecc, uint32_t ob_dtc0ecc, uint32_t ob_dtc1ecc);
<b>Function descriptions</b>	configure TCM ECC option byte
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_itcmecc</b>	ITCM ECC function enable bit
OB_ITCMECCEN_DISABLE	disabled ITCM ECC function
OB_ITCMECCEN_ENABLE	enabled ITCM ECC function
<b>Input parameter{in}</b>	
<b>ob_dtc0ecc</b>	DTCM0 ECC function enable bit
OB_DTCM0ECCEN_DISABLE	disabled DTCM0 ECC function
OB_DTCM0ECCEN_ENABLE	enabled DTCM0 ECC function
<b>Input parameter{in}</b>	
<b>ob_dtc1ecc</b>	DTCM1 ECC function enable bit
OB_DTCM1ECCEN_DISABLE	disabled DTCM1 ECC function
OB_DTCM1ECCEN_ENABLE	enabled DTCM1 ECC function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* configure TCM ECC option byte */
```

```
fmc_state = fmc_state_enum ob_tcm_ecc_config(OB_ITCMECCEN_ENABLE, OB_DTC
M0ECCEN_DISABLE, OB_DTCM1ECCEN_DISABLE);
```

### ob\_iospeed\_optimize\_config

The description of ob\_iospeed\_optimize\_config is shown as below:

**Table 3-476. Function ob\_iospeed\_optimize\_config**

<b>Function name</b>	ob_iospeed_optimize_config
<b>Function prototype</b>	fmc_state_enum ob_iospeed_optimize_config(uint32_t ob_iospeed_op);
<b>Function descriptions</b>	configure I/O speed optimization option byte
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_iospeed_op</b>	configure I/O speed optimization, high-speed at low-voltage enable bit
<i>OB_IOSPDOPEN_DISABLE</i>	disabled I/O speed optimization
<i>OB_IOSPDOPEN_ENABLE</i>	enabled I/O speed optimization
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* disabled I/O speed optimization */
```

```
fmc_state = fmc_state_enum ob_iospeed_optimize_config(OB_IOSPDOPEN_DISABLE);
```

### ob\_tcm\_shared\_ram\_config

The description of ob\_tcm\_shared\_ram\_config is shown as below:

**Table 3-477. Function ob\_tcm\_shared\_ram\_config**

<b>Function name</b>	ob_tcm_shared_ram_config
<b>Function prototype</b>	fmc_state_enum ob_tcm_shared_ram_config(uint32_t itcm_shared_ram_size, uint32_t dtcm_shared_ram_size);
<b>Function descriptions</b>	configure option byte TCM shared RAM size
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-

Input parameter{in}	
<b>itcm_shared_ram_size</b>	ITCM shared RAM size
<i>OB_ITCM_SHARED_RAM_0KB</i>	ITCM shared RAM size is 0KB
<i>OB_ITCM_SHARED_RAM_64KB</i>	ITCM shared RAM size is 64KB
<i>OB_ITCM_SHARED_RAM_128KB</i>	ITCM shared RAM size is 128KB
<i>OB_ITCM_SHARED_RAM_256KB</i>	ITCM shared RAM size is 256KB
<i>OB_ITCM_SHARED_RAM_512KB</i>	ITCM shared RAM size is 512KB
Input parameter{in}	
<b>dtcm_shared_ram_size</b>	DTCM shared RAM size
<i>OB_DTCM_SHARED_RAM_0KB</i>	DTCM shared RAM size is .KB
<i>OB_DTCM_SHARED_RAM_64KB</i>	DTCM shared RAM size is 64KB
<i>OB_DTCM_SHARED_RAM_128KB</i>	DTCM shared RAM size is 128KB
<i>OB_DTCM_SHARED_RAM_256KB</i>	DTCM shared RAM size is 256KB
<i>OB_DTCM_SHARED_RAM_512KB</i>	DTCM shared RAM size is 512KB
Output parameter{out}	
-	-
Return value	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;
```

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* configure option byte TCM shared RAM size */
```

```
fmc_state = fmc_state_enum ob_tcm_shared_ram_config(OB_ITCM_SHARED_RAM_64KB,
OB_DTCM_SHARED_RAM_64KB);
```

### ob\_data\_program

The description of ob\_data\_program is shown as below:

Table 3-478. Function ob\_data\_program

Function name	ob_data_program
Function prototype	fmc_state_enum ob_data_program(uint16_t ob_data);
Function descriptions	modify option byte DATA
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 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* modify option byte DATA */

fmc_state = ob_data_program (0x1234);
```

### ob\_boot\_address\_config

The description of ob\_boot\_address\_config is shown as below:

Table 3-479. Function ob\_boot\_address\_config

Function name	ob_boot_address_config
Function prototype	fmc_state_enum ob_boot_address_config(uint8_t boot_pin, uint16_t boot_address);
Function descriptions	configure boot address
Precondition	ob_unlock
The called functions	-
Input parameter{in}	
boot_pin	configure I/O speed optimization, high-speed at low-voltage enable bit
BOOT_PIN_0	boot pin value is 0
BOOT_PIN_1	boot pin value is 1
Input parameter{in}	
boot_address	specify the MSB of boot address
Output parameter{out}	
-	-
Return value	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* disabled I/O speed optimization */

fmc_state = fmc_state_enum ob_boot_address_config(BOOT_PIN_1, 0x1FF0)
```

### ob\_dcrp\_area\_config

The description of ob\_dcrp\_area\_config is shown as below:

**Table 3-480. Function ob\_dcrp\_area\_config**

<b>Function name</b>	ob_dcrp_area_config
<b>Function prototype</b>	fmc_state_enum ob_dcrp_area_config(uint32_t dcrp_eren, uint32_t dcrp_start, uint32_t dcrp_end);
<b>Function descriptions</b>	configure DCRP area
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>dcrp_eren</b>	DCRP area erase enable bit
OB_DCRP_AREA_ERASE_DISABLE	DCRP area erase disable
OB_DCRP_AREA_ERASE_ENABLE	DCRP area erase enable
<b>Input parameter{in}</b>	
<b>dcrp_start</b>	DCRP area start address
<b>Input parameter{in}</b>	
<b>dcrp_end</b>	DCRP area end address
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* configure DCRP area */

fmc_state = fmc_state_enum ob_dcrp_area_config(OB_DCRP_AREA_ERASE_ENABLE,
0x10, 0x1F)
```

## ob\_secure\_area\_config

The description of ob\_secure\_area\_config is shown as below:

**Table 3-481. Function ob\_secure\_area\_config**

<b>Function name</b>	ob_secure_area_config
<b>Function prototype</b>	fmc_state_enum ob_secure_area_config(uint32_t scr_eren, uint32_t scr_start, uint32_t scr_end);
<b>Function descriptions</b>	configure secure-access area
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>scr_eren</b>	secure-access area erase enable bit
OB_SCR_AREA_ERASE_DISABLE	secure-access area erase disable
OB_SCR_AREA_ERASE_ENABLE	secure-access area erase enable
<b>Input parameter{in}</b>	
<b>scr_start</b>	secure-access area start address
<b>Input parameter{in}</b>	
<b>dcrp_end</b>	secure-access area end address
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_state_enum fmc_state;

fmc_unlock();

ob_unlock();

/* configure secure-access area */

fmc_state = fmc_state_enum ob_secure_area_config(OB_SCR_AREA_ERASE_ENABLE,
0x10, 0x1F)
```

## ob\_write\_protection\_enable

The description of ob\_write\_protection\_enable is shown as below:

**Table 3-482. Function ob\_write\_protection\_enable**

<b>Function name</b>	ob_write_protection_enable
<b>Function prototype</b>	fmc_state_enum ob_write_protection_enable(uint32_t ob_wp);
<b>Function descriptions</b>	enable option bytes sector erase/program protection
<b>Precondition</b>	ob_unlock

The called functions	-
Input parameter{in}	
<b>ob_wp</b>	specify sector to be erase/program protected
<i>OB_WP_0</i>	erase/program protect sector 0 ~ sector 15
<i>OB_WP_1</i>	erase/program protect sector 16 ~ sector 31
<i>OB_WP_2</i>	erase/program protect sector 32 ~ sector 47
<i>OB_WP_3</i>	erase/program protect sector 48 ~ sector 63
<i>OB_WP_4</i>	erase/program protect sector 64 ~ sector 79
<i>OB_WP_5</i>	erase/program protect sector 80 ~ sector 95
<i>OB_WP_6</i>	erase/program protect sector 96 ~ sector 111
<i>OB_WP_7</i>	erase/program protect sector 112 ~ sector 127
<i>OB_WP_8</i>	erase/program protect sector 128 ~ sector 143
<i>OB_WP_9</i>	erase/program protect sector 144 ~ sector 159
<i>OB_WP_10</i>	erase/program protect sector 160 ~ sector 175
<i>OB_WP_11</i>	erase/program protect sector 176 ~ sector 191
<i>OB_WP_12</i>	erase/program protect sector 192 ~ sector 207
<i>OB_WP_13</i>	erase/program protect sector 208 ~ sector 223
<i>OB_WP_14</i>	erase/program protect sector 224 ~ sector 239
<i>OB_WP_15</i>	erase/program protect sector 240 ~ sector 255
<i>OB_WP_16</i>	erase/program protect sector 256 ~ sector 383
<i>OB_WP_17</i>	erase/program protect sector 384 ~ sector 511
<i>OB_WP_18</i>	erase/program protect sector 512 ~ sector 639
<i>OB_WP_19</i>	erase/program protect sector 640 ~ sector 767
<i>OB_WP_20</i>	erase/program protect sector 768 ~ sector 895
<i>OB_WP_21</i>	erase/program protect sector 896 ~ sector 959
<i>OB_WP_ALL</i>	all sectors
Output parameter{out}	
-	-
Return value	
<b>fmc_state_enum</b>	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* enable sector 80 ~ sector 95 erase/program protection */
```

```
fmc_state_enum fmc_state = ob_write_protection_enable(OB_WP_5);
```

### ob\_write\_protection\_disable

The description of ob\_write\_protection\_disable is shown as below:

**Table 3-483. Function ob\_write\_protection\_disable**

Function name	ob_write_protection_disable
---------------	-----------------------------



<b>Function prototype</b>	fmc_state_enum ob_write_protection_disable(uint32_t ob_wp);
<b>Function descriptions</b>	disable option bytes sector erase/program protection
<b>Precondition</b>	ob_unlock
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ob_wp</b>	specify sector to be erase/program protected
OB_WP_0	erase/program protect sector 0 ~ sector 15
OB_WP_1	erase/program protect sector 16 ~ sector 31
OB_WP_2	erase/program protect sector 32 ~ sector 47
OB_WP_3	erase/program protect sector 48 ~ sector 63
OB_WP_4	erase/program protect sector 64 ~ sector 79
OB_WP_5	erase/program protect sector 80 ~ sector 95
OB_WP_6	erase/program protect sector 96 ~ sector 111
OB_WP_7	erase/program protect sector 112 ~ sector 127
OB_WP_8	erase/program protect sector 128 ~ sector 143
OB_WP_9	erase/program protect sector 144 ~ sector 159
OB_WP_10	erase/program protect sector 160 ~ sector 175
OB_WP_11	erase/program protect sector 176 ~ sector 191
OB_WP_12	erase/program protect sector 192 ~ sector 207
OB_WP_13	erase/program protect sector 208 ~ sector 223
OB_WP_14	erase/program protect sector 224 ~ sector 239
OB_WP_15	erase/program protect sector 240 ~ sector 255
OB_WP_16	erase/program protect sector 256 ~ sector 383
OB_WP_17	erase/program protect sector 384 ~ sector 511
OB_WP_18	erase/program protect sector 512 ~ sector 639
OB_WP_19	erase/program protect sector 640 ~ sector 767
OB_WP_20	erase/program protect sector 768 ~ sector 895
OB_WP_21	erase/program protect sector 896 ~ sector 959
OB_WP_ALL	all sectors
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
fmc_state_enum	state of FMC, refer to <a href="#">Table 3-453. Enum fmc_state_enum</a>

Example:

```
fmc_unlock();
```

```
ob_unlock();
```

```
/* disable sector 80 ~ sector 95 erase/program protection */
```

```
fmc_state_enum fmc_state = ob_write_protection_disable(OB_WP_5);
```

## ob\_secure\_mode\_get

The description of ob\_secure\_mode\_get is shown as below:

**Table 3-484. Function ob\_secure\_mode\_get**

<b>Function name</b>	ob_secure_mode_get
<b>Function prototype</b>	FlagStatus ob_secure_mode_get(void);
<b>Function descriptions</b>	get the option byte secure access mode
<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 option byte secure access mode */
```

```
FlagStatus flag = ob_secure_mode_get( );
```

## ob\_security\_protection\_flag\_get

The description of ob\_security\_protection\_flag\_get is shown as below:

**Table 3-485. Function ob\_secure\_mode\_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 option byte security protection level
<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 option byte security protection level */
```

```
FlagStatus flag = ob_security_protection_flag_get();
```

## ob\_bor\_threshold\_get

The description of ob\_bor\_threshold\_get is shown as below:

Table 3-486. Function `ob_bor_threshold_get`

Function name	<code>ob_bor_threshold_get</code>
Function prototype	<code>uint32_t ob_bor_threshold_get(void);</code>
Function descriptions	get the option byte BOR threshold value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<code>uint32_t</code>	the BOR threshold value
<code>OB_BOR_TH_VALUE3</code>	BOR threshold value 3
<code>OB_BOR_TH_VALUE2</code>	BOR threshold value 2
<code>OB_BOR_TH_VALUE1</code>	BOR threshold value 1
<code>OB_BOR_TH_OFF</code>	no BOR function

Example:

```
/* get the BOR threshold value */
```

```
uint32_t user = ob_bor_threshold_get();
```

### **ob\_low\_power\_get**

The description of `ob_low_power_get` is shown as below:

Table 3-487. Function `ob_low_power_get`

Function name	<code>ob_low_power_get</code>
Function prototype	<code>void ob_low_power_get(uint32_t *fwdgt, uint32_t *deepsleep, uint32_t *standby, uint32_t *fwdg_suspend_deepsleep, uint32_t *fwdg_suspend_standby);</code>
Function descriptions	get low power related option byte
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
<code>fwdgt</code>	watchdog option
<code>OB_FWDGT_SW</code>	software free watchdog
<code>OB_FWDGT_HW</code>	hardware free watchdog
Output parameter{out}	
<code>deepsleep</code>	deepsleep reset option
<code>OB_DEEPSLEEP_NRS</code> <code>T</code>	no reset when entering deepsleep mode
<code>OB_DEEPSLEEP_RST</code>	generate a reset instead of entering deepsleep mode

Output parameter{out}	
<b>standby</b>	standby reset option
<i>OB_STDBY_NRST</i>	no reset when entering standby mode
<i>OB_STDBY_RST</i>	generate a reset instead of entering standby mode
Output parameter{out}	
<b>fwdg_suspend_deepsleep</b>	FWDG suspend status in deep-sleep mode option
<i>OB_DPSLP_FWDGT_SUSPEND</i>	free watchdog is suspended in deepsleep mode
<i>OB_DPSLP_FWDGT_RUN</i>	free watchdog is running in deepsleep mode
Output parameter{out}	
<b>fwdg_suspend_standby</b>	FWDG suspend status in standby mode option
<i>OB_STDBY_FWDGT_SUSPEND</i>	free watchdog is suspended in standby mode
<i>OB_STDBY_FWDGT_RUN</i>	free watchdog is running in standby mode
Return value	
-	-

Example:

```
/* get low power related option byte */
```

```
uint32_t fwdgt_value, deepsleep_value, standby_value, fwdg_suspend_deepsleep_value,
fwdg_suspend_standby_value;
```

```
ob_low_power_get(&fwdgt_value, &deepsleep_value, &standby_value, &fwdg_suspend_deepsleep_value,
&fwdg_suspend_standby_value);
```

### ob\_tcm\_ecc\_get

The description of ob\_tcm\_ecc\_get is shown as below:

**Table 3-488. Function ob\_tcm\_ecc\_get**

<b>Function name</b>	ob_tcm_ecc_get
<b>Function prototype</b>	void ob_tcm_ecc_get(uint32_t *itcm_ecc_option, uint32_t *dcm0ecc_option, uint32_t *dcm1ecc_option);
<b>Function descriptions</b>	get TCM ECC configuration
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
<b>itcm_ecc_option</b>	ITCM ECC function enable option

<i>OB_ITCMECCEN_DISABLE</i>	disabled ITCM ECC function
<i>OB_ITCMECCEN_ENABLE</i>	enabled ITCM ECC function
<b>Output parameter{out}</b>	
<b>dtcm0ecc_option</b>	DTCM0 ECC function enable bit
<i>OB_DTCM0ECCEN_DISABLE</i>	disabled DTCM0 ECC function
<i>OB_DTCM0ECCEN_ENABLE</i>	enabled DTCM0 ECC function
<b>Output parameter{out}</b>	
<b>ob_dtcm1ecc</b>	DTCM1 ECC function enable bit
<i>OB_DTCM1ECCEN_DISABLE</i>	disabled DTCM1 ECC function
<i>OB_DTCM1ECCEN_ENABLE</i>	enabled DTCM1 ECC function
<b>Return value</b>	
-	-

Example:

```
/* get TCM ECC configuration */
```

```
uint32_t itcmecc_option_value, dtcm0ecc_option_value, dtcm1ecc_option_value;
```

```
ob_tcm_ecc_get(&itcmecc_option_value, &dtcm0ecc_option_value, &dtcm1ecc_option_value);
```

### ob\_iospeed\_option\_get

The description of ob\_iospeed\_option\_get is shown as below:

**Table 3-489. Function ob\_secure\_mode\_get**

<b>Function name</b>	ob_iospeed_option_get
<b>Function prototype</b>	FlagStatus ob_iospeed_option_get(void);
<b>Function descriptions</b>	get IO speed optimize configuration
<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 IO speed optimize configuration */
```

```
FlagStatus flag = ob_iospeed_option_get();
```

### ob\_itcm\_shared\_ram\_size\_get

The description of ob\_itcm\_shared\_ram\_size\_get is shown as below:

**Table 3-490. Function ob\_itcm\_shared\_ram\_size\_get**

<b>Function name</b>	ob_itcm_shared_ram_size_get
<b>Function prototype</b>	uint32_t ob_itcm_shared_ram_size_get (void);
<b>Function descriptions</b>	get the option byte ITCM shared RAM size
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	ITCM shared RAM size

Example:

```
/* get ITCM shared RAM size */
```

```
uint32_t data = ob_itcm_shared_ram_size_get();
```

### ob\_dtcn\_shared\_ram\_size\_get

The description of ob\_dtcn\_shared\_ram\_size\_get is shown as below:

**Table 3-491. Function ob\_dtcn\_shared\_ram\_size\_get**

<b>Function name</b>	ob_dtcn_shared_ram_size_get
<b>Function prototype</b>	uint32_t ob_dtcn_shared_ram_size_get (void);
<b>Function descriptions</b>	get the option byte DTCM shared RAM size
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	DTCM shared RAM size

Example:

```
/* get DTCM shared RAM size */
```

```
uint32_t data = ob_dtcn_shared_ram_size_get();
```

## ob\_data\_get

The description of ob\_data\_get is shown as below:

**Table 3-492. 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 user data value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint16_t	option bytes user data

Example:

```
/* get the value of FMC option bytes DATA in FMC_OBSTAT1 register */
uint16_t data = ob_data_get();
```

## ob\_boot\_address\_get

The description of ob\_boot\_address\_get is shown as below:

**Table 3-493. Function ob\_boot\_address\_get**

<b>Function name</b>	ob_boot_address_get
<b>Function prototype</b>	uint32_t ob_boot_address_get(uint8_t boot_pin);
<b>Function descriptions</b>	get boot address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>boot_pin</b>	boot pin configuration
<i>BOOT_PIN_0</i>	boot pin value is 0
<i>BOOT_PIN_1</i>	boot pin value is 1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	boot address

Example:

```
/* get boot address */
uint32_t bootaddr = ob_boot_address_get(BOOT_PIN_0);
```

## ob\_dcrp\_area\_get

The description of ob\_dcrp\_area\_get is shown as below:

**Table 3-494. Function ob\_dcrp\_area\_get**

<b>Function name</b>	ob_dcrp_area_get
<b>Function prototype</b>	uint8_t ob_dcrp_area_get(uint32_t *dcrp_erase_option, uint32_t *dcrp_area_start_addr, uint32_t *dcrp_area_end_addr);
<b>Function descriptions</b>	get DCRP area configuration
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>dcrp_erase_option</b>	DCRP area erase option
<i>OB_DCRP_AREA_ERASE_DISABLE</i>	DCRP area erase disable
<i>OB_DCRP_AREA_ERASE_ENABLE</i>	DCRP area erase enable
<b>Output parameter{out}</b>	
<b>dcrp_start_addr</b>	DCRP area start address
<b>Output parameter{out}</b>	
<b>dcrp_end_addr</b>	DCRP area end address
<b>Return value</b>	
<b>uint8_t</b>	INVLD_AREA_ADDRESS or VLD_AREA_ADDRESS

Example:

```
/* get DCRP area configuration */
```

```
uint32_t dcrp_erase_opt, dcrp_startaddr, dcrp_endaddr;
```

```
uint8_t flag = ob_dcrp_area_get(&dcrp_erase_opt, &dcrp_startaddr, &dcrp_endaddr);
```

## ob\_secure\_area\_get

The description of ob\_secure\_area\_get is shown as below:

**Table 3-495. Function ob\_secure\_area\_get**

<b>Function name</b>	ob_secure_area_get
<b>Function prototype</b>	uint8_t ob_secure_area_get(uint32_t *secure_area_option, uint32_t *scr_area_start_addr, uint32_t *scr_area_end_addr);
<b>Function descriptions</b>	get secure-access area configuration
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-



Output parameter{out}	
<b>secure_erase_option</b>	secure-access area erase option
<i>OB_SCR_AREA_ERASE_DISABLE</i>	secure-access area erase disable
<i>OB_SCR_AREA_ERASE_ENABLE</i>	secure-access area erase enable
Output parameter{out}	
<b>scr_area_start_addr</b>	secure-access area start address
Output parameter{out}	
<b>scr_area_end_addr</b>	secure-access area end address
Return value	
<b>uint8_t</b>	INVLD_AREA_ADDRESS or VLD_AREA_ADDRESS

Example:

```
/* get secure-access area configuration */
```

```
uint32_t scr_erase_opt, scr_startaddr, scr_endaddr;
```

```
uint8_t flag = ob_dcrp_area_get(&scr_erase_opt, &scr_startaddr, &scr_endaddr);
```

### ob\_write\_protection\_get

The description of ob\_write\_protection\_get is shown as below:

**Table 3-496. 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 option byte erase/program protection
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<b>uint32_t</b>	the FMC erase/program protection option byte value(0 - 0x3FFFFFFF)

Example:

```
/* get the option byte erase/program protection */
```

```
uint32_t wp = ob_write_protection_get();
```

### fmc\_no\_rtdec\_config

The description of fmc\_no\_rtdec\_config is shown as below:

Table 3-497. Function `fmc_no_rtdec_config`

Function name	<code>fmc_no_rtdec_config</code>
Function prototype	<code>fmc_state_enum fmc_no_rtdec_config(uint32_t nodec_area_start, uint32_t nodec_area_end);</code>
Function descriptions	configure NO-RTDEC area
Precondition	<code>fmc_unlock</code>
The called functions	-
Input parameter{in}	
<code>nodec_area_start</code>	no rtdec area start address
<code>nodec_area_end</code>	no rtdec area end address
Output parameter{out}	
-	-
Return value	
<code>fmc_state_enum</code>	state of FMC, refer to <a href="#">Table 3-453. Enum <code>fmc_state_enum</code></a>

Example:

```
/* configure NO-RTDEC area */
```

```
fmc_state_enum fmc_state = fmc_no_rtdec_config (2U, 4U);
```

### `fmc_aes_iv_config`

The description of `fmc_aes_iv_config` is shown as below:

Table 3-498. Function `fmc_aes_iv_config`

Function name	<code>fmc_aes_iv_config</code>
Function prototype	<code>fmc_state_enum fmc_aes_iv_config(uint32_t *aes_iv);</code>
Function descriptions	configure aes initialization vector
Precondition	<code>fmc_unlock</code>
The called functions	-
Input parameter{in}	
<code>aes_iv</code>	high 96 bits of AES initialization vector
Output parameter{out}	
-	-
Return value	
<code>fmc_state_enum</code>	state of FMC, refer to <a href="#">Table 3-453. Enum <code>fmc_state_enum</code></a>

Example:

```
/* configure NO-RTDEC area */
```

```
uint32_t aes_high96[3] = {0x11111111U, 0x22222222U, 0x33333333U};
```

```
fmc_state_enum fmc_state = fmc_aes_iv_config(2U, 4U);
```

## fmc\_flash\_ecc\_get

The description of fmc\_flash\_ecc\_get is shown as below:

**Table 3-499. Function fmc\_flash\_ecc\_get**

<b>Function name</b>	fmc_flash_ecc_get
<b>Function prototype</b>	FlagStatus fmc_flash_ecc_get(void);
<b>Function descriptions</b>	get Flash ECC function enable flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the value of flash ECC function enable flag */
```

```
FlagStatus flag = fmc_flash_ecc_get();
```

## fmc\_no\_rtdec\_get

The description of fmc\_no\_rtdec\_get is shown as below:

**Table 3-500. Function fmc\_no\_rtdec\_get**

<b>Function name</b>	fmc_no_rtdec_get
<b>Function prototype</b>	void fmc_no_rtdec_get(uint32_t *nodec_area_start, uint32_t *nodec_area_end);
<b>Function descriptions</b>	get NO-RTDEC area
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>nodec_area_start</b>	no rtdec area start address
<b>Output parameter{out}</b>	
<b>nodec_area_end</b>	no rtdec area end address
<b>Return value</b>	
-	-

Example:

```
/* get NO-RTDEC area */
```

```
uint32_t nodec_startaddr, nodec_endaddr;
```

```
fmc_no_rtdec_get(&nodec_startaddr, &nodec_endaddr);
```

### fmc\_aes\_iv\_get

The description of fmc\_aes\_iv\_get is shown as below:

**Table 3-501. Function fmc\_aes\_iv\_get**

<b>Function name</b>	fmc_aes_iv_get
<b>Function prototype</b>	void fmc_aes_iv_get(uint32_t *aes_iv);
<b>Function descriptions</b>	get AES initialization vector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
aes_iv	high 96 bits of AES initialization vector
<b>Return value</b>	
-	-

Example:

```
/* get AES initialization vector */
```

```
uint32_t aes_iv[3];
```

```
fmc_aes_iv_get(&aes_iv);
```

### fmc\_pid\_get

The description of fmc\_pid\_get is shown as below:

**Table 3-502. Function fmc\_pid\_get**

<b>Function name</b>	fmc_pid_get
<b>Function prototype</b>	void fmc_pid_get(uint32_t *pid);
<b>Function descriptions</b>	get product ID
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
pid	product ID
<b>Return value</b>	
-	-

Example:

```
/* get product ID */
```

```
uint32_t pid[2];
```

```
fmc_pid_get(&pid);
```

## fmc\_flag\_get

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

**Table 3-503. 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 3-454. Enum fmc_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get FMC flag */
```

```
FlagStatus flag = fmc_flag_get(FMC_FLAG_WPERR);
```

## fmc\_flag\_clear

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

**Table 3-504. 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 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 3-454. Enum fmc_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear RPERR flag */
```

```
fmc_flag_clear(FMC_FLAG_RPERR);
```

## fmc\_interrupt\_enable

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

**Table 3-505. 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>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FMC interrupt, refer to <a href="#">Table 3-456. Enum fmc_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FMC end interrupt */
fmc_interrupt_enable(FMC_INT_END);
```

## fmc\_interrupt\_disable

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

**Table 3-506. 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>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	FMC interrupt, refer to <a href="#">Table 3-456. Enum fmc_interrupt_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FMC end interrupt */
fmc_interrupt_disable(FMC_INT_END);
```

## fmc\_interrupt\_flag\_get

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

**Table 3-507. 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 status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fmc_int_flag</b>	FMC interrupt flag, refer to <a href="#">Table 3-455. Enum fmc_interrupt_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* check FMC program sequence error flag is set or not */
```

```
FlagStatus flag = fmc_interrupt_flag_get(FMC_INT_FLAG_PGSERR);
```

### fmc\_interrupt\_flag\_clear

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

**Table 3-508. 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 status
<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 3-455. Enum fmc_interrupt_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear FMC program sequence error flag */
```

```
fmc_interrupt_flag_get(FMC_INT_FLAG_PGSERR);
```

## 3.17. FWDGT

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

chapter [3.17.1](#). The FWDGT firmware functions are introduced in chapter [3.17.2](#).

### 3.17.1. Descriptions of Peripheral registers

FWDGT registers are listed in the table shown as below:

**Table 3-509. FWDGT Registers**

Registers	Descriptions
FWDGT_CTL	control register
FWDGT_PSC	prescaler register
FWDGT_RLD	reload register
FWDGT_STAT	status register
FWDGT_WND	window register

### 3.17.2. Descriptions of Peripheral functions

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

**Table 3-510. FWDGT firmware function**

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

#### fwdgt\_write\_enable

The description of fwdgt\_write\_enable is shown as below:

**Table 3-511. Function fwdgt\_write\_enable**

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



-	-
Return value	
-	-

Example:

```
/* enable write access to FWDGT_PSC and FWDGT_RLD and FWDGT_WND */
```

```
 fwdgt_write_enable();
```

### **fwdgt\_write\_disable**

The description of fwdgt\_write\_disable is shown as below:

**Table 3-512. 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 3-513. 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 3-514. 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 3-515. Function fwdgt\_reload\_value\_config**

<b>Function name</b>	fwdgt_reload_value_config
<b>Function prototype</b>	ErrStatus fwdgt_reload_value_config(uint16_t reload_value);
<b>Function descriptions</b>	configure the FWDGT counter reload value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>reload_value</b>	reload_value, specify reload value(0x0000 - 0x0FFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

<b>ErrStatus</b>	ERROR / SUCCESS
------------------	-----------------

Example:

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

ErrStatus flag;

```
flag = fwdgt_reload_value_config(0xFFFF);
```

### fwdgt\_window\_value\_config

The description of fwdgt\_window\_value\_config is shown as below:

**Table 3-516. Function fwdgt\_window\_value\_config**

<b>Function name</b>	fwdgt_window_value_config
<b>Function prototype</b>	ErrStatus fwdgt_window_value_config(uint16_t window_value);
<b>Function descriptions</b>	configure the FWDGT counter window value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>window_value</b>	window_value, specify window value(0x0000 - 0x0FFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR / SUCCESS

Example:

```
/* set FWDGT window value to 0xFFFF */
```

ErrStatus flag;

```
flag = fwdgt_window_value_config(0xFFFF);
```

### fwdgt\_counter\_reload

The description of fwdgt\_counter\_reload is shown as below:

**Table 3-517. Function fwdgt\_counter\_reload**

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

Return value	
-	-

Example:

```
/* reload FWDGT counter */
```

```
fwdgt_counter_reload();
```

### fwdgt\_config

The description of fwdgt\_config is shown as below:

**Table 3-518. Function fwdgt\_config**

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

Example:

```
/* confiure FWDGT counter clock: 32KHz(IRC32K) / 64 = 0.5 KHz */
```

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

### fwdgt\_flag\_get

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

**Table 3-519. Function fwdgt\_flag\_get**

Function name	fwdgt_flag_get
Function prototype	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);
```

## 3.18. GPIO

GPIO is used to implement logic input/output functions for the devices. The GPIO registers are listed in chapter [3.18.1](#), the GPIO firmware functions are introduced in chapter [3.18.2](#).

### 3.18.1. Descriptions of Peripheral registers

GPIO registers are listed in the table shown as below:

**Table 3-520. GPIO Registers**

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

Registers	Descriptions
GPIO_IFTP	GPIO input filtering type register

### 3.18.2. Descriptions of Peripheral functions

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

**Table 3-521. GPIO firmware function**

Function name	Function description
gpio_deinit	reset GPIO port
gpio_mode_set	set GPIO mode
gpio_output_options_set	set GPIO output type and speed
gpio_bit_set	set GPIO pin bit
gpio_bit_reset	reset GPIO pin bit
gpio_bit_write	write data to the specified GPIO pin
gpio_port_write	write data to the specified GPIO port
gpio_input_filter_set	set GPIO input filter
gpio_input_bit_get	get GPIO pin input status
gpio_input_port_get	get GPIO port input status
gpio_output_bit_get	get GPIO pin output status
gpio_output_port_get	get GPIO port output status
gpio_af_set	set GPIO alternate function
gpio_pin_lock	lock GPIO pin bit
gpio_bit_toggle	toggle GPIO pin status
gpio_port_toggle	toggle GPIO port status

#### gpio\_deinit

The description of gpio\_deinit is shown as below:

**Table 3-522. Function gpio\_deinit**

<b>Function name</b>	gpio_deinit
<b>Function prototype</b>	void gpio_deinit(uint32_t gpio_periph);
<b>Function descriptions</b>	reset GPIO port
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
<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 3-523. 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
GPIOx	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
<b>Input parameter{in}</b>	
<b>mode</b>	gpio pin mode
GPIO_MODE_INPUT	input mode
GPIO_MODE_OUTPUT	output mode
GPIO_MODE_AF	alternate function mode
GPIO_MODE_ANALOG	analog mode
<b>Input parameter{in}</b>	
<b>pull_up_down</b>	gpio pin with pull-up or pull-down resistor
GPIO_PUPD_NONE	floating mode, no pull-up and pull-down resistors
GPIO_PUPD_PULLUP	with pull-up resistor
GPIO_PUPD_PULLDOWN	with pull-down resistor
<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:

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

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

## gpio\_output\_options\_set

The description of gpio\_output\_options\_set is shown as below:

**Table 3-524. Function gpio\_output\_options\_set**

<b>Function name</b>	gpio_output_options_set
<b>Function prototype</b>	void gpio_output_options_set(uint32_t gpio_periph, uint8_t otype, uint32_t speed, uint32_t pin);
<b>Function descriptions</b>	set GPIO output type and speed
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
<b>Input parameter{in}</b>	
<b>otype</b>	gpio pin output mode
<i>GPIO_OTYPE_PP</i>	push pull mode
<i>GPIO_OTYPE_OD</i>	open drain mode
<b>Input parameter{in}</b>	
<b>speed</b>	gpio pin output max speed
<i>GPIO_OSPEED_12MHZ</i>	output max speed 12MHz
<i>GPIO_OSPEED_60MHZ</i>	output max speed 60MHz
<i>GPIO_OSPEED_85MHZ</i>	output max speed 85MHz
<i>GPIO_OSPEED_100_20MHZ</i>	output max speed 100/220MHz
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

```
gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_12MHZ,
GPIO_PIN_0);
```



## gpio\_bit\_set

The description of gpio\_bit\_set is shown as below:

**Table 3-525. 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,H,J,K)
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set PA0*/
```

```
gpio_bit_set(GPIOA, GPIO_PIN_0);
```

## gpio\_bit\_reset

The description of gpio\_bit\_reset is shown as below:

**Table 3-526. 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,H,J,K)
<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	
-	-

Example:

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

```
gpio_bit_set(GPIOA, GPIO_PIN_0);
```

## gpio\_bit\_write

The description of gpio\_bit\_write is shown as below:

**Table 3-527. 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,H,J,K)
<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 3-528. 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,H,J,K)
<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\_filter\_set

The description of gpio\_input\_filter\_set is shown as below:

**Table 3-529. Function gpio\_input\_filter\_set**

<b>Function name</b>	gpio_input_filter_set
<b>Function prototype</b>	void gpio_input_filter_set(uint32_t gpio_periph, uint8_t speriod, uint32_t iftype, uint32_t pin);
<b>Function descriptions</b>	set GPIO input filter
<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,H,J,K)
<b>Input parameter{in}</b>	
<b>speriod</b>	gpio pin input sample period
<i>GPIO_ISPERIOD(x)</i>	period (x = 0 ~ 255)
<b>Input parameter{in}</b>	
<b>iftype</b>	gpio pin input filtering type
<i>GPIO_IFTYPE_SYNC</i>	synchronization type
<i>GPIO_IFTYPE_3_SAMPLE</i>	filter 3 samples
<i>GPIO_IFTYPE_6_SAMPLE</i>	filter 6 samples
<i>GPIO_IFTYPE_ASYNC</i>	asynchronous type
<b>Input parameter{in}</b>	

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

Example:

```
/* set GPIO input filter */
```

```
gpio_input_filter_set(GPIOA, GPIO_IPERIOD(100), GPIO_IFTYPE_SYNC);
```

### gpio\_input\_bit\_get

The description of gpio\_input\_bit\_get is shown as below:

**Table 3-530. 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,H,J,K)
<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 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 3-531. 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,H,J,K)
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 3-532. Function gpio\_output\_bit\_get

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

Example:

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

```
FlagStatus bit_state;
```

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

## gpio\_output\_port\_get

The description of gpio\_output\_port\_get is shown as below:

**Table 3-533. 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,H,J,K)
<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 3-534. Function gpio\_af\_set**

<b>Function name</b>	gpio_af_set
<b>Function prototype</b>	void gpio_af_set(uint32_t gpio_periph, uint32_t alt_func_num, uint32_t pin);
<b>Function descriptions</b>	set GPIO alternate function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_periph</b>	GPIO port
<i>GPIOx</i>	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
<b>Input parameter{in}</b>	
<b>alt_func_num</b>	GPIO pin af function, please refer to specific device datasheet
<i>GPIO_AF_0</i>	SYSTEM, TIMER40, TIMER41, TIMER42, TIMER43, TIMER44
<i>GPIO_AF_1</i>	TIMER0, TIMER1, TIMER15, TIMER16, EXMC
<i>GPIO_AF_2</i>	TIMER2, TIMER3, TIMER4, TIMER7, TIMER14, CAN2, EXMC

<i>GPIO_AF_3</i>	<i>TIMER7, TIMER9, EDOUT, EXMC, HPDF, OSPIM</i>
<i>GPIO_AF_4</i>	<i>TIMER14, I2C0, I2C1, I2C2, I2C3, USART0, HPDF, OSPIM</i>
<i>GPIO_AF_5</i>	<i>SPI0, SPI1, SPI2, SPI3, SPI4, SPI5, CAN2</i>
<i>GPIO_AF_6</i>	<i>UART3, SPI2, I2C3, HPDF, EDOUT, OSPIM</i>
<i>GPIO_AF_7</i>	<i>USART0, USART1, USART2, USART5, UART6, TIMER40, TIMER41, TIMER42, TIMER43, SPI1, SPI2, SPI5, USBHS1</i>
<i>GPIO_AF_8</i>	<i>UART3, UART4, UART7, SPI5, TIMER44, USBHS1</i>
<i>GPIO_AF_9</i>	<i>TRGSEL, CAN0, CAN1, TLI, OPSIM, EXMC</i>
<i>GPIO_AF_10</i>	<i>CMP, USBHS0, OPSIM, EXMC</i>
<i>GPIO_AF_11</i>	<i>CMP, UART6, EXMC, HPDF, I2C3, OPSIM</i>
<i>GPIO_AF_12</i>	<i>TIMER0, EXMC, OPSIM, CMP, USBHS1</i>
<i>GPIO_AF_13</i>	<i>TRGSEL, COMP0, CMP, TIMER22</i>
<i>GPIO_AF_14</i>	<i>UART4, TIMER23</i>
<i>GPIO_AF_15</i>	<i>EVENTOUT</i>
<b>Input parameter{in}</b>	
<b>pin</b>	GPIO pin
<i>GPIO_PIN_x</i>	GPIO_PIN_x(x=0..15)
<i>GPIO_PIN_ALL</i>	All pins
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set PA0 alternate function 0 */
```

```
gpio_af_set(GPIOA, GPIO_AF_0, GPIO_PIN_0);
```

## gpio\_pin\_lock

The description of gpio\_pin\_lock is shown as below:

**Table 3-535. 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,H,J,K)
<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

Output parameter{out}	
-	-
Return value	
-	-

Example:

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

## gpio\_bit\_toggle

The description of gpio\_bit\_toggle is shown as below:

**Table 3-536. Function gpio\_bit\_toggle**

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

Example:

```
/* toggle PA0 */
gpio_bit_toggle(GPIOA, GPIO_PIN_0);
```

## gpio\_port\_toggle

The description of gpio\_port\_toggle is shown as below:

**Table 3-537. Function gpio\_port\_toggle**

Function name	gpio_port_toggle
Function prototype	void gpio_port_toggle(uint32_t gpio_periph);
Function descriptions	toggle GPIO port status
Precondition	-



The called functions	-
Input parameter{in}	
gpio_periph	GPIO port
GPIOx	GPIOx(x = A,B,C,D,E,F,G,H,J,K)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* toggle GPIOA*/
gpio_port_toggle(GPIOA);
```

## 3.19. HPDF

A high performance digital filter module (HPDF) for external sigma delta ( $\Sigma$ - $\Delta$ ) modulator is integrated in GD32H75E. The HPDF registers are listed in chapter [3.19.1](#), the HPDF firmware functions are introduced in chapter [3.19.2](#).

### 3.19.1. Descriptions of Peripheral registers

HPDF registers are listed in the table shown as below:

**Table 3-538. HPDF Registers**

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

Registers	Descriptions
HPDF_FLTyEMMAX	HPDF Filter y extremes monitor maximum register
HPDF_FLTyEMMIN	HPDF Filter y extremes monitor minimum register
HPDF_FLTyCT	HPDF Filter y conversion timer register

### 3.19.2. Descriptions of Peripheral functions

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

**Table 3-539. HPDF firmware function**

Function name	Function description
hpdf_deinit	reset HPDF
hpdf_channel_struct_para_init	initialize the parameters of HPDF channel struct with the default values
hpdf_filter_struct_para_init	initialize the parameters of HPDF filter struct with the default values
hpdf_rc_struct_para_init	initialize the parameters of regular conversion struct with the default values
hpdf_ic_struct_para_init	initialize the parameters of inserted conversion struct with the default values
hpdf_enable	enable the HPDF module globally
hpdf_disable	disable the HPDF module globally
hpdf_channel_init	initialize the HPDF channel
hpdf_filter_init	initialize the HPDF filter
hpdf_rc_init	initialize the regular conversion
hpdf_ic_init	initialize the inserted conversion
hpdf_clock_output_config	configure serial output clock
hpdf_clock_output_source_config	configure serial clock output source
hpdf_clock_output_duty_mode_disable	disable serial clock output duty mode
hpdf_clock_output_duty_mode_enable	enable serial clock output duty mode
hpdf_clock_output_divider_config	configure serial clock output divider
hpdf_channel_enable	enable channel
hpdf_channel_disable	disable channel
hpdf_spi_clock_source_config	configure SPI clock source
hpdf_serial_interface_type_config	configure serial interface type
hpdf_malfunction_monitor_disable	disable malfunction monitor
hpdf_malfunction_monitor_enable	enable malfunction monitor
hpdf_clock_loss_disable	disable clock loss detector
hpdf_clock_loss_enable	enable clock loss detector
hpdf_channel_pin_redirection_disable	disable channel inputs pins redirection
hpdf_channel_pin_redirection_enable	enable channel inputs pins redirection
hpdf_channel_mux_config	configure channel multiplexer select input data source
hpdf_data_pack_mode_config	configure data packing mode
hpdf_data_right_bit_shift_config	configure data right bit-shift

Function name	Function description
hpdf_calibration_offset_config	configure calibration offset
hpdf_malfunction_break_signal_config	configure malfunction monitor break signal
hpdf_malfunction_counter_config	configure malfunction monitor counter threshold
hpdf_write_parallel_data_standard_mode	write the parallel data on standard mode of data packing
hpdf_write_parallel_data_interleaved_mode	write the parallel data on interleaved mode of data packing
hpdf_write_parallel_data_dual_mode	write the parallel data on dual mode of data packing
hpdf_pulse_skip_update	update the number of pulses to skip
hpdf_pulse_skip_read	read the number of pulses to skip
hpdf_filter_enable	enable filter
hpdf_filter_disable	disable filter
hpdf_filter_config	configure sinc filter order and oversample
hpdf_integrator_oversample	configure integrator oversampling rate
hpdf_threshold_monitor_filter_config	configure threshold monitor filter order and oversample
hpdf_threshold_monitor_filter_read_data	read the threshold monitor filter data
hpdf_threshold_monitor_fast_mode_disable	disable threshold monitor fast mode
hpdf_threshold_monitor_fast_mode_enable	enable threshold monitor fast mode
hpdf_threshold_monitor_channel	configure threshold monitor channel
hpdf_threshold_monitor_high_threshold	configure threshold monitor high threshold value
hpdf_threshold_monitor_low_threshold	configure threshold monitor low threshold value
hpdf_high_threshold_break_signal	configure threshold monitor high threshold event break signal
hpdf_low_threshold_break_signal	configure threshold monitor low threshold event break signal
hpdf_extremes_monitor_channel	configure extremes monitor channel
hpdf_extremes_monitor_maximum_get	get the extremes monitor maximum value
hpdf_extremes_monitor_minimum_get	get the extremes monitor minimum value
hpdf_conversion_time_get	get the conversion timer value
hpdf_rc_continuous_disable	disable regular conversions continuous mode
hpdf_rc_continuous_enable	enable regular conversions continuous mode
hpdf_rc_start_by_software	start regular channel conversion by software
hpdf_rc_syn_disable	disable regular conversion synchronously
hpdf_rc_syn_enable	enable regular conversion synchronously
hpdf_rc_dma_disable	disable regular conversion DMA channel
hpdf_rc_dma_enable	enable regular conversion DMA channel
hpdf_rc_channel_config	configure regular conversion channel
hpdf_rc_fast_mode_disable	disable regular conversion fast conversion mode
hpdf_rc_fast_mode_enable	enable regular conversion fast conversion mode
hpdf_rc_data_get	get the regular conversion data
hpdf_rc_channel_get	get the channel of regular channel most recently

Function name	Function description
	converted
hpdf_ic_start_by_software	start inserted channel conversion by software
hpdf_ic_syn_disable	disable inserted conversion synchronously
hpdf_ic_syn_enable	enable inserted conversion synchronously
hpdf_ic_dma_disable	disable inserted conversion DMA channel
hpdf_ic_dma_enable	enable inserted conversion DMA channel
hpdf_ic_scan_mode_disable	disable scan conversion mode
hpdf_ic_scan_mode_enable	enable scan conversion mode
hpdf_ic_trigger_signal_disable	disable inserted conversions trigger signal
hpdf_ic_trigger_signal_config	configure inserted conversions trigger signal and trigger edge
hpdf_ic_channel_config	configure inserted group conversions channel
hpdf_ic_data_get	get the inserted conversions data
hpdf_ic_channel_get	get the channel of inserted group channel most recently converted
hpdf_flag_get	get the HPDF flags
hpdf_flag_clear	clear the HPDF flags
hpdf_interrupt_enable	enable HPDF interrupt
hpdf_interrupt_disable	disable HPDF interrupt
hpdf_interrupt_flag_get	get the HPDF interrupt flags
hpdf_interrupt_flag_clear	clear the HPDF interrupt flags

### Structure hpdf\_channel\_parameter\_struct

**Table 3-540. Structure hpdf\_channel\_parameter\_struct**

Member name	Function description
data_packing_mode	ata packing mode for HPDF_CHxPDI register
channel_muxlexer	channel multiplexer select input data source
channel_pin_select	channel inputs pins selection
ck_loss_detector	clock loss detector
malfunction_monitor	malfunction monitor
spi_ck_source	SPI clock source select
serial_interface	serial interface type
calibration_offset	24-bit calibration offset
right_bit_shift	data right bit-shift
tm_filter	threshold monitor Sinc filter order selection
tm_filter_oversample	threshold monitor filter oversampling rate
mm_break_signal	malfunction monitor break signal distribution
mm_counter_threshold	malfunction monitor counter threshold
plsk_value	the number of serial input samples that will be skipped

## Structure hpdf\_filter\_parameter\_struct

**Table 3-541. Structure hpdf\_filter\_parameter\_struct**

Member name	Function description
tm_fast_mode	threshold monitor fast mode
tm_channel	threshold monitor channel
tm_high_threshold	threshold monitor high threshold
tm_low_threshold	threshold monitor low threshold value
extreme_monitor_channel	extremes monitor channel
sinc_filter	sinc filter order
sinc_oversample	sinc filter oversampling rate
integrator_oversample	integrator oversampling rate
ht_break_signal	high threshold event break signal distribution
lt_break_signal	low threshold event break signal distribution

## Structure hpdf\_rc\_parameter\_struct

**Table 3-542. Structure hpdf\_rc\_parameter\_struct**

Member name	Function description
fast_mode	fast conversion mode enable for regular conversions
rsc_channel	regular conversion channel
rcdmaen	DMA channel enabled to read data for the regular conversion
rcsyn	regular conversion synchronously
continuous_mode	regular conversions continuous mode

## Structure hpdf\_ic\_parameter\_struct

**Table 3-543. Structure hpdf\_ic\_parameter\_struct**

Member name	Function description
trigger_edge	inserted conversions trigger edge
trigger_signal	inserted conversions trigger signal
icdmaen	DMA channel enabled to read data for the inserted channel group
scmod	scan conversion mode of inserted conversions
icsyn	inserted conversion synchronously
ic_channel_group	inserted channel group selection

## Enum hpdf\_channel\_enum

**Table 3-544. Enum hpdf\_channel\_enum**

enum name	enum description
CHANNEL0	HPDF channel0
CHANNEL1	HPDF channel1
CHANNEL2	HPDF channel2
CHANNEL3	HPDF channel3
CHANNEL4	HPDF channel4

enum name	enum description
CHANNEL5	HPDF channel5
CHANNEL6	HPDF channel6
CHANNEL7	HPDF channel7

### Enum hpdf\_filter\_enum

**Table 3-545. Enum hpdf\_filter\_enum**

enum name	enum description
FLT0	HPDF filter0
FLT1	HPDF filter1
FLT2	HPDF filter2
FLT3	HPDF filter3

### Enum hpdf\_flag\_enum

**Table 3-546. Enum hpdf\_flag\_enum**

enum name	enum description
HPDF_FLAG_FLTy _ICEF	inserted conversion end flag
HPDF_FLAG_FLTy _RCEF	regular conversion end flag
HPDF_FLAG_FLTy _ICDOF	inserted conversion overflow flag
HPDF_FLAG_FLTy _RCDOF	regular conversion overflow flag
HPDF_FLAG_FLTy _TMEOF	threshold monitor event occurred flag
HPDF_FLAG_FLTy _ICPF	inserted conversion in progress flag
HPDF_FLAG_FLTy _RCPF	regular conversion in progress flag
HPDF_FLAG_FLT0 _CKLF0	clock loss on channel 0 flag
HPDF_FLAG_FLT0 _CKLF1	clock loss on channel 1 flag
HPDF_FLAG_FLT0 _CKLF2	clock loss on channel 2 flag
HPDF_FLAG_FLT0 _CKLF3	clock loss on channel 3 flag
HPDF_FLAG_FLT0 _CKLF4	clock loss on channel 4 flag
HPDF_FLAG_FLT0 _CKLF5	clock loss on channel 5 flag

enum name	enum description
HPDF_FLAG_FLT0_CKLF6	clock loss on channel 6 flag
HPDF_FLAG_FLT0_CKLF7	clock loss on channel 7 flag
HPDF_FLAG_FLT0_MMF0	malfunction event occurred on channel 0 flag
HPDF_FLAG_FLT0_MMF1	malfunction event occurred on channel 1 flag
HPDF_FLAG_FLT0_MMF2	malfunction event occurred on channel 2 flag
HPDF_FLAG_FLT0_MMF3	malfunction event occurred on channel 3 flag
HPDF_FLAG_FLT0_MMF4	malfunction event occurred on channel 4 flag
HPDF_FLAG_FLT0_MMF5	malfunction event occurred on channel 5 flag
HPDF_FLAG_FLT0_MMF6	malfunction event occurred on channel 6 flag
HPDF_FLAG_FLT0_MMF7	malfunction event occurred on channel 7 flag
HPDF_FLAG_FLTy_RCHPDT	regular channel pending data
HPDF_FLAG_FLTy_LTF0	threshold monitor low threshold on channel 0 flag
HPDF_FLAG_FLTy_LTF1	threshold monitor low threshold on channel 1 flag
HPDF_FLAG_FLTy_LTF2	threshold monitor low threshold on channel 2 flag
HPDF_FLAG_FLTy_LTF3	threshold monitor low threshold on channel 3 flag
HPDF_FLAG_FLTy_LTF4	threshold monitor low threshold on channel 4 flag
HPDF_FLAG_FLTy_LTF5	threshold monitor low threshold on channel 5 flag
HPDF_FLAG_FLTy_LTF6	threshold monitor low threshold on channel 6 flag
HPDF_FLAG_FLTy_LTF7	threshold monitor low threshold on channel 7 flag
HPDF_FLAG_FLTy_HTF0	threshold monitor high threshold on channel 0 flag
HPDF_FLAG_FLTy_HTF1	threshold monitor high threshold on channel 1 flag

enum name	enum description
HPDF_FLAG_FLTy_HTF2	threshold monitor high threshold on channel 2 flag
HPDF_FLAG_FLTy_HTF3	threshold monitor high threshold on channel 3 flag
HPDF_FLAG_FLTy_HTF4	threshold monitor high threshold on channel 4 flag
HPDF_FLAG_FLTy_HTF5	threshold monitor high threshold on channel 5 flag
HPDF_FLAG_FLTy_HTF6	threshold monitor high threshold on channel 6 flag
HPDF_FLAG_FLTy_HTF7	threshold monitor high threshold on channel 7 flag

### Enum hpdf\_interrput\_flag\_enum

**Table 3-547. Enum hpdf\_interrput\_flag\_enum**

enum name	enum description
HPDF_INT_FLAG_FLTy_ICEF	inserted conversion end interrupt flag
HPDF_INT_FLAG_FLTy_RCEF	regular conversion end interruptflag
HPDF_INT_FLAG_FLTy_ICDOF	inserted conversion overflow interrupt flag
HPDF_INT_FLAG_FLTy_RCDOF	regular conversion overflow interrupt flag
HPDF_INT_FLAG_FLTy_TMEOF	threshold monitor event occurred interrupt flag
HPDF_INT_FLAG_FLT0_CKLF0	clock loss on channel 0 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF1	clock loss on channel 1 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF2	clock loss on channel 2 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF3	clock loss on channel 3 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF4	clock loss on channel 4 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF5	clock loss on channel 5 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF6	clock loss on channel 6 interrupt flag
HPDF_INT_FLAG_FLT0_CKLF7	clock loss on channel 7 interrupt flag



enum name	enum description
HPDF_INT_FLAG_FLT0_MMF0	malfunction monitor detection on channel 0 interrupt flag
HPDF_INT_FLAG_FLT0_MMF1	malfunction monitor detection on channel 1 interrupt flag
HPDF_INT_FLAG_FLT0_MMF2	malfunction monitor detection on channel 2 interrupt flag
HPDF_INT_FLAG_FLT0_MMF3	malfunction monitor detection on channel 3 interrupt flag
HPDF_INT_FLAG_FLT0_MMF4	malfunction monitor detection on channel 4 interrupt flag
HPDF_INT_FLAG_FLT0_MMF5	malfunction monitor detection on channel 5 interrupt flag
HPDF_INT_FLAG_FLT0_MMF6	malfunction monitor detection on channel 6 interrupt flag
HPDF_INT_FLAG_FLT0_MMF7	malfunction monitor detection on channel 7 interrupt flag

### Enum hpdf\_interrput\_enum

**Table 3-548. Enum hpdf\_interrput\_enum**

enum name	enum description
HPDF_INT_FLTy_I CEIE	inserted conversion end interrupt enable
HPDF_INT_FLTy_R CEIE	regular conversion end interrupt enable
HPDF_INT_FLTy_I CDOIE	inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_R CDOIE	regular conversion data overflow interrupt enable
HPDF_INT_FLTy_T MIE	threshold monitor interrupt enable
HPDF_INT_FLT0_M MIE	malfunction monitor interrupt enable
HPDF_INT_FLT0_C KLIE	clock loss interrupt enable

### hpdf\_deinit

The description of hpdf\_deinit is shown as below:

**Table 3-549. Function hpdf\_deinit**

<b>Function name</b>	hpdf_deinit
<b>Function prototype</b>	void hpdf_deinit(void);
<b>Function descriptions</b>	reset HPDF

<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset HPDF */
```

```
hpdf_deinit();
```

### hpdf\_channel\_struct\_para\_init

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

**Table 3-550. Function hpdf\_channel\_struct\_para\_init**

<b>Function name</b>	hpdf_channel_struct_para_init
<b>Function prototype</b>	void hpdf_channel_struct_para_init(hpdf_channel_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of HPDF channel struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
*init_struct	the initialized struct needed to initialize HPDF channel, refer to <a href="#">Table 3-540. Structure hpdf_channel_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of HPDF channel */
```

```
hpdf_channel_parameter_struct hpdf_channel_init_struct;
```

```
hpdf_channel_struct_para_init(&hpdf_channel_init_struct);
```

### hpdf\_filter\_struct\_para\_init

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

**Table 3-551. Function hpdf\_filter\_struct\_para\_init**

<b>Function name</b>	hpdf_filter_struct_para_init
<b>Function prototype</b>	void hpdf_filter_struct_para_init(hpdf_filter_parameter_struct* init_struct);

<b>Function descriptions</b>	initialize the parameters of HPDF filter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF filter, refer to <a href="#">Table 3-541. Structure hpdf_filter_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of HPDF filter */
hpdf_filter_parameter_struct hpdf_filter_init_struct;
hpdf_filter_struct_para_init(&hpdf_filter_init_struct);
```

### hpdf\_rc\_struct\_para\_init

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

**Table 3-552. Function hpdf\_rc\_struct\_para\_init**

<b>Function name</b>	hpdf_rc_struct_para_init
<b>Function prototype</b>	void hpdf_rc_struct_para_init(hpdf_rc_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of regular conversion struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF regular conversion, refer to <a href="#">Table 3-542. Structure hpdf_rc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of regular conversion */
hpdf_rc_parameter_struct hpdf_rc_init_struct;
hpdf_rc_struct_para_init(&hpdf_rc_init_struct);
```

### hpdf\_ic\_struct\_para\_init

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

**Table 3-553. Function hpdf\_ic\_struct\_para\_init**

<b>Function name</b>	hpdf_ic_struct_para_init
<b>Function prototype</b>	void hpdf_ic_struct_para_init(hpdf_ic_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of inserted conversion struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF inserted conversion, refer to <a href="#">Table 3-543. Structure hpdf_ic_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of inserted conversion */
hpdf_ic_parameter_struct hpdf_ic_init_struct;
hpdf_ic_struct_para_init(&hpdf_ic_init_struct);
```

### hpdf\_enable

The description of hpdf\_enable is shown as below:

**Table 3-554. Function hpdf\_enable**

<b>Function name</b>	hpdf_enable
<b>Function prototype</b>	void hpdf_enable(void);
<b>Function descriptions</b>	enable the HPDF module globally
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the HPDF module */
hpdf_enable();
```

### hpdf\_disable

The description of hpdf\_disable is shown as below:

**Table 3-555. Function hpdf\_disable**

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

Example:

```
/* disable the HPDF module */
```

```
hpdf_disable();
```

### hpdf\_channel\_init

The description of hpdf\_channel\_init is shown as below:

**Table 3-556. Function hpdf\_channel\_init**

<b>Function name</b>	hpdf_channel_init
<b>Function prototype</b>	void hpdf_channel_init(hpdf_channel_enum channelx, hpdf_channel_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the HPDF channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF channel, refer to <a href="#">Table 3-540. Structure hpdf_channel_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the HPDF channel0 */
```

```
hpdf_channel_parameter_struct hpdf_channel_init_struct;
```

```
/* initialize HPDF channel0 */
```

```

hpdf_channel_init_struct.data_packing_mode = DPM_STANDARD_MODE;

hpdf_channel_init_struct.malfunction_monitor = MM_ENABLE;

hpdf_channel_init_struct.spi_ck_source = EXTERNAL_CKIN;

hpdf_channel_init_struct.channel_muxlexer = SERIAL_INPUT;

hpdf_channel_init_struct.serial_interface = SPI_RISING_EDGE;

hpdf_channel_init_struct.calibration_offset = 0;

hpdf_channel_init_struct.right_bit_shift = 0;

hpdf_channel_init_struct.mm_counter_threshold = 110;

hpdf_channel_init_struct.plsk_value = 0;

hpdf_channel_init(CHANNEL0, &hpdf_channel_init_struct);

```

### hpdf\_filter\_init

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

**Table 3-557. Function hpdf\_filter\_init**

<b>Function name</b>	hpdf_filter_init
<b>Function prototype</b>	void hpdf_filter_init(hpdf_filter_enum filtery, hpdf_filter_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the HPDF filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..7)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF filter, refer to <a href="#">Table 3-541. Structure hpdf_filter_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* initialize the HPDF filter0 */

hpdf_filter_parameter_struct hpdf_filter_init_struct;

hpdf_filter_init_struct.tm_fast_mode = TMFM_DISABLE;

hpdf_filter_init_struct.tm_channel = TMCHEN_CHANNEL0;

```

```

hpdf_filter_init_struct.tm_high_threshold = tm_high_val;

hpdf_filter_init_struct.tm_low_threshold = tm_low_val;

hpdf_filter_init_struct.extreme_monitor_channel = EM_CHANNEL0;

hpdf_filter_init_struct.sinc_filter = FLT_SINC3;

hpdf_filter_init_struct.sinc_oversample = FLT_OVER_SAMPLE_32;

hpdf_filter_init_struct.integrator_oversample = INTEGRATOR_BYPASS;

hpdf_filter_init(FLT0, &hpdf_filter_init_struct);

```

## hpdf\_rc\_init

The description of hpdf\_rc\_init is shown as below:

**Table 3-558. Function hpdf\_rc\_init**

<b>Function name</b>	hpdf_rc_init
<b>Function prototype</b>	void hpdf_rc_init(hpdf_filter_enum filtery, hpdf_rc_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the regular conversion
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF regular conversion, refer to <a href="#">Table 3-542. Structure hpdf_rc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* initialize regular conversion of the HPDF fliter0 */

hpdf_rc_parameter_struct hpdf_rc_init_struct;

hpdf_rc_init_struct.fast_mode = FAST_DISABLE;

hpdf_rc_init_struct.rcs_channel = RCS_CHANNEL0;

hpdf_rc_init_struct.rcdmaen = RCDMAEN_ENABLE;

hpdf_rc_init_struct.continuous_mode = RCCM_ENABLE;

hpdf_rc_init(FLT0, &hpdf_rc_init_struct);

```

## hpdf\_ic\_init

The description of hpdf\_ic\_init is shown as below:

**Table 3-559. Function hpdf\_ic\_init**

<b>Function name</b>	hpdf_ic_init
<b>Function prototype</b>	void hpdf_ic_init(hpdf_filter_enum filtery, hpdf_ic_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the inserted conversion
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>*init_struct</b>	the initialized struct needed to initialize HPDF inserted conversion, refer to <a href="#">Table 3-543. Structure hpdf_ic_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* initialize inserted conversion of the HPDF filter0 */

hpdf_ic_parameter_struct hpdf_ic_init_struct;

hpdf_ic_init_struct.ic_channel_group = IGCSEL_CHANNEL0;

hpdf_ic_init_struct.scmmod = SCMOD_ENABLE;

hpdf_ic_init_struct.icdmaen = ICDMAEN_ENABLE;

hpdf_ic_init_struct.icsyn = ICSYN_DISABLE ;

hpdf_ic_init(FLT0, &hpdf_ic_init_struct);

```

## hpdf\_clock\_output\_config

The description of hpdf\_clock\_output\_config is shown as below:

**Table 3-560. Function hpdf\_clock\_output\_config**

<b>Function name</b>	hpdf_clock_output_config
<b>Function prototype</b>	void hpdf_clock_output_config(uint32_t source, uint8_t divider, uint32_t mode);
<b>Function descriptions</b>	configure serial output clock
<b>Precondition</b>	disable HPDF
<b>The called functions</b>	-



Input parameter{in}	
<b>source</b>	the HPDF serial clock output source
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from system clock
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from audio clock
Input parameter{in}	
<b>divider</b>	serial clock output divider(0-255)
Input parameter{in}	
<b>mode</b>	serial clock output duty mode
<i>CKOUTDM_DISABLE</i>	disable serial clock output duty mode
<i>CKOUTDM_ENABLE</i>	enable serial clock output duty mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure serial clock output */
```

```
hpdf_clock_output_config(SERIAL_SYSTEM_CLK, 175, CKOUTDM_ENABLE);
```

### hpdf\_clock\_output\_source\_config

The description of hpdf\_clock\_output\_source\_config is shown as below:

**Table 3-561. Function hpdf\_clock\_output\_source\_config**

<b>Function name</b>	hpdf_clock_output_source_config
<b>Function prototype</b>	void hpdf_clock_output_source_config(uint32_t source);
<b>Function descriptions</b>	configure serial clock output source
<b>Precondition</b>	disable HPDF
<b>The called functions</b>	-
Input parameter{in}	
<b>source</b>	the HPDF serial clock output source
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from system clock
<i>SERIAL_SYSTEM_CLK</i>	serial clock output source is from audio clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure serial clock output source */
```

```
hpdf_clock_output_config(SERIAL_SYSTEM_CLK);
```

## hpdf\_clock\_output\_duty\_mode\_disable

The description of hpdf\_clock\_output\_duty\_mode\_disable is shown as below:

**Table 3-562. Function hpdf\_clock\_output\_duty\_mode\_disable**

<b>Function name</b>	hpdf_clock_output_duty_mode_disable
<b>Function prototype</b>	void hpdf_clock_output_duty_mode_disable(void);
<b>Function descriptions</b>	disable serial clock output duty mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable serial clock output duty mode*/
hpdf_clock_output_duty_mode_disable();
```

## hpdf\_clock\_output\_duty\_mode\_enable

The description of hpdf\_clock\_output\_duty\_mode\_enable is shown as below:

**Table 3-563. Function hpdf\_clock\_output\_duty\_mode\_enable**

<b>Function name</b>	hpdf_clock_output_duty_mode_enable
<b>Function prototype</b>	void hpdf_clock_output_duty_mode_enable(void);
<b>Function descriptions</b>	enable serial clock output duty 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 serial clock output duty mode*/
hpdf_clock_output_duty_mode_enable();
```

## hpdf\_clock\_output\_divider\_config

The description of hpdf\_clock\_output\_divider\_config is shown as below:

**Table 3-564. Function hpdf\_clock\_output\_divider\_config**

<b>Function name</b>	hpdf_clock_output_divider_config
<b>Function prototype</b>	void hpdf_clock_output_divider_config(uint8_t divider);
<b>Function descriptions</b>	configure serial clock output divider
<b>Precondition</b>	disable HPDF
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>divider</b>	serial clock output divider(0-255)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure serial clock output divider */
```

```
hpdf_clock_output_divider_config (255);
```

### hpdf\_channel\_enable

The description of hpdf\_channel\_enable is shown as below:

**Table 3-565. Function hpdf\_channel\_enable**

<b>Function name</b>	hpdf_channel_enable
<b>Function prototype</b>	void hpdf_channel_enable(hpdf_channel_enum channelx);
<b>Function descriptions</b>	enable channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable channel0 */
```

```
hpdf_channel_enable(CHANNEL0);
```

### hpdf\_channel\_disable

The description of hpdf\_channel\_disable is shown as below:

Table 3-566. Function `hpdf_channel_disable`

Function name	<code>hpdf_channel_disable</code>
Function prototype	<code>void hpdf_channel_disable(hpdf_channel_enum channelx);</code>
Function descriptions	disable channel
Precondition	-
The called functions	-
Input parameter{in}	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum <i>hpdf_channel_enum</i></a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable channel0 */
hpdf_channel_disable(CHANNEL0);
```

### `hpdf_spi_clock_source_config`

The description of `hpdf_spi_clock_source_config` is shown as below:

Table 3-567. Function `hpdf_spi_clock_source_config`

Function name	<code>hpdf_spi_clock_source_config</code>
Function prototype	<code>void hpdf_spi_clock_source_config(hpdf_channel_enum channelx, uint32_t clock_source);</code>
Function descriptions	configure SPI clock source
Precondition	-
The called functions	-
Input parameter{in}	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum <i>hpdf_channel_enum</i></a>
Input parameter{in}	
<b>clock_source</b>	SPI clock source
<i>EXTERNAL_CKIN</i>	external input clock
<i>INTERNAL_CKOUT</i>	internal CKOUT clock
<i>HALF_CKOUT_FALLING_EDGE</i>	internal CKOUT clock, sampling point on each second CKOUT falling edge
<i>HALF_CKOUT_RISING_EDGE</i>	internal CKOUT clock, sampling point on each second CKOUT rising edge
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure external input clock as SPI clock source */
hpdf_spi_clock_source_config(CHANNEL0, EXTERNAL_CKIN);
```

### hpdf\_serial\_interface\_type\_config

The description of hpdf\_serial\_interface\_type\_config is shown as below:

**Table 3-568. Function hpdf\_serial\_interface\_type\_config**

<b>Function name</b>	hpdf_serial_interface_type_config
<b>Function prototype</b>	void hpdf_serial_interface_type_config(hpdf_channel_enum channelx, uint32_t type);
<b>Function descriptions</b>	configure serial interface type
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>type</b>	serial interface type
SPI_RISING_EDGE	SPI interface, sample data on rising edge
SPI_FALLING_EDGE	SPI interface, sample data on rising edge
MANCHESTER_CODE 0	Manchester coded input: rising edge = logic 0, falling edge = logic 1
MANCHESTER_CODE 1	Manchester coded input: rising edge = logic 1, falling edge = logic 0
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure external input clock as SPI clock source */
hpdf_spi_clock_source_config(CHANNEL0, EXTERNAL_CKIN);
```

### hpdf\_malfunction\_monitor\_disable

The description of hpdf\_malfunction\_monitor\_disable is shown as below:

**Table 3-569. Function hpdf\_malfunction\_monitor\_disable**

<b>Function name</b>	hpdf_malfunction_monitor_disable
<b>Function prototype</b>	void hpdf_malfunction_monitor_disable(hpdf_channel_enum channelx);
<b>Function descriptions</b>	disable malfunction monitor
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable malfunction monitor */
hpdf_malfunction_monitor_disable(CHANNEL0);
```

### hpdf\_malfunction\_monitor\_enable

The description of hpdf\_malfunction\_monitor\_enable is shown as below:

**Table 3-570. Function hpdf\_malfunction\_monitor\_enable**

Function name	hpdf_malfunction_monitor_enable
Function prototype	void hpdf_malfunction_monitor_enable(hpdf_channel_enum channelx);
Function descriptions	enable malfunction monitor
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable malfunction monitor */
hpdf_malfunction_monitor_enable(CHANNEL1);
```

### hpdf\_clock\_loss\_disable

The description of hpdf\_clock\_loss\_disable is shown as below:

**Table 3-571. Function hpdf\_clock\_loss\_disable**

Function name	hpdf_clock_loss_disable
Function prototype	void hpdf_clock_loss_disable(hpdf_channel_enum channelx);
Function descriptions	disable clock loss detector
Precondition	disable CHANNELx(x=0..7)

The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable clock loss detector */
hpdf_clock_loss_disable(CHANNEL0);
```

### hpdf\_clock\_loss\_enable

The description of hpdf\_clock\_loss\_enable is shown as below:

**Table 3-572. Function hpdf\_clock\_loss\_enable**

Function name	hpdf_clock_loss_enable
Function prototype	void hpdf_clock_loss_enable(hpdf_channel_enum channelx);
Function descriptions	enable clock loss detector
Precondition	disable CHANNELx(x=0..7)
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable clock loss detector */
hpdf_clock_loss_enable(CHANNEL0);
```

### hpdf\_channel\_pin\_redirection\_disable

The description of hpdf\_channel\_pin\_redirection\_disable is shown as below:

**Table 3-573. Function hpdf\_channel\_pin\_redirection\_disable**

Function name	hpdf_channel_pin_redirection_disable
Function prototype	void hpdf_channel_pin_redirection_disable(hpdf_channel_enum channelx);
Function descriptions	disable channel inputs pins redirection
Precondition	disable CHANNELx(x=0..7)

The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable channel0 inputs pins redirection */
```

```
hpdf_channel_pin_redirection_disable(CHANNEL0);
```

### hpdf\_channel\_pin\_redirection\_enable

The description of hpdf\_channel\_pin\_redirection\_enable is shown as below:

**Table 3-574. Function hpdf\_channel\_pin\_redirection\_enable**

Function name	hpdf_channel_pin_redirection_enable
Function prototype	void hpdf_channel_pin_redirection_enable(hpdf_channel_enum channelx);
Function descriptions	enable channel inputs pins redirection
Precondition	disable CHANNELx(x=0..7)
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable channel0 inputs pins redirection */
```

```
hpdf_channel_pin_redirection_enable(CHANNEL0);
```

### hpdf\_channel\_mux\_config

The description of hpdf\_channel\_mux\_config is shown as below:

**Table 3-575. Function hpdf\_channel\_mux\_config**

Function name	hpdf_channel_mux_config
Function prototype	void hpdf_channel_mux_config(hpdf_channel_enum channelx, uint32_t data_source);
Function descriptions	configure channel multiplexer select input data source



<b>Precondition</b>	disable CHANNELx(x=0..7)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>data_source</b>	input data source
SERIAL_INPUT	input data source is taken from serial inputs
INTERNAL_INPUT	input data source is taken from internal HPDF_CHxPDI register
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure channel multiplexer select input data source */
hpdf_channel_mux_config(CHANNEL0, SERIAL_INPUT);
```

### hpdf\_data\_pack\_mode\_config

The description of hpdf\_data\_pack\_mode\_config is shown as below:

**Table 3-576. Function hpdf\_data\_pack\_mode\_config**

<b>Function name</b>	hpdf_data_pack_mode_config
<b>Function prototype</b>	void hpdf_data_pack_mode_config(hpdf_channel_enum channelx, uint32_t mode);
<b>Function descriptions</b>	configure data packing mode
<b>Precondition</b>	disable CHANNELx(x=0..7)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>mode</b>	parallel data packing mode
DPM_STANDARD_MODE	standard mode
DPM_INTERLEAVED_MODE	interleaved mode
DPM_DUAL_MODE	dual mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data packing mode */
```

```
hpdf_data_pack_mode_config(CHANNEL0, DPM_STANDARD_MODE);
```

### hpdf\_data\_right\_bit\_shift\_config

The description of hpdf\_data\_right\_bit\_shift\_config is shown as below:

**Table 3-577. Function hpdf\_data\_right\_bit\_shift\_config**

<b>Function name</b>	hpdf_data_right_bit_shift_config
<b>Function prototype</b>	void hpdf_data_right_bit_shift_config(hpdf_channel_enum channelx, uint8_t right_shift);
<b>Function descriptions</b>	configure data right bit-shift
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>right_shift</b>	the number of bits that determine the right shift(0-31)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data right bit-shift */
```

```
hpdf_data_right_bit_shift_config(CHANNEL0, 5);
```

### hpdf\_calibration\_offset\_config

The description of hpdf\_calibration\_offset\_config is shown as below:

**Table 3-578. Function hpdf\_calibration\_offset\_config**

<b>Function name</b>	hpdf_calibration_offset_config
<b>Function prototype</b>	void hpdf_calibration_offset_config(hpdf_channel_enum channelx, int32_t offset);
<b>Function descriptions</b>	configure calibration offset
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	

<b>offset</b>	24-bit calibration offset, must be in (-8388608~8388607)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure calibration offset */
```

```
hpdf_calibration_offset_config (CHANNEL0, -255);
```

### hpdf\_malfunction\_break\_signal\_config

The description of hpdf\_malfunction\_break\_signal\_config is shown as below:

**Table 3-579. Function hpdf\_malfunction\_break\_signal\_config**

<b>Function name</b>	hpdf_malfunction_break_signal_config
<b>Function prototype</b>	void hpdf_malfunction_break_signal_config(hpdf_channel_enum channelx, uint32_t break_signal);
<b>Function descriptions</b>	configure malfunction monitor break signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>break_signal</b>	malfunction monitor break signal distribution
NO_MM_BREAK	break signal is not distributed to malfunction monitor on channel
MM_BREAK0	break signal 0 is distributed to malfunction monitor on channel
MM_BREAK1	break signal 1 is distributed to malfunction monitor on channel
MM_BREAK2	break signal 2 is distributed to malfunction monitor on channel
MM_BREAK3	break signal 3 is distributed to malfunction monitor on channel
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure break signal is distributed to malfunction monitor on channel0 */
```

```
hpdf_data_pack_mode_config(CHANNEL0, MM_BREAK0);
```

### hpdf\_malfunction\_counter\_config

The description of hpdf\_malfunction\_counter\_config is shown as below:

**Table 3-580. Function hpdf\_malfunction\_counter\_config**

<b>Function name</b>	hpdf_malfunction_counter_config
<b>Function prototype</b>	void hpdf_malfunction_counter_config(hpdf_channel_enum channelx, uint8_t threshold);
<b>Function descriptions</b>	configure malfunction monitor counter threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>threshold</b>	malfunction monitor counter threshold(0-255)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure malfunction monitor counter threshold */
hpdf_malfunction_counter_config(CHANNEL0, 255);
```

### hpdf\_write\_parallel\_data\_standard\_mode

The description of hpdf\_write\_parallel\_data\_standard\_mode is shown as below:

**Table 3-581. Function hpdf\_write\_parallel\_data\_standard\_mode**

<b>Function name</b>	hpdf_write_parallel_data_standard_mode
<b>Function prototype</b>	void hpdf_write_parallel_data_standard_mode(hpdf_channel_enum channelx, int16_t data);
<b>Function descriptions</b>	write the parallel data on standard mode of data packing
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>data</b>	the parallel data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write the parallel data on standard mode of data packing */
```

```
hpdf_write_parallel_data_standard_mode(CHANNEL0, 0xEFFF);
```

### hpdf\_write\_parallel\_data\_interleaved\_mode

The description of hpdf\_write\_parallel\_data\_interleaved\_mode is shown as below:

**Table 3-582. Function hpdf\_write\_parallel\_data\_interleaved\_mode**

<b>Function name</b>	hpdf_write_parallel_data_interleaved_mode
<b>Function prototype</b>	void hpdf_write_parallel_data_interleaved_mode(hpdf_channel_enum channelx, int32_t data);
<b>Function descriptions</b>	write the parallel data on interleaved mode of data packing
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>data</b>	the parallel data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write the parallel data on interleaved mode of data packing */
```

```
hpdf_write_parallel_data_interleaved_mode(CHANNEL0, 0xEFFFFFFF);
```

### hpdf\_write\_parallel\_data\_dual\_mode

The description of hpdf\_write\_parallel\_data\_dual\_mode is shown as below:

**Table 3-583. Function hpdf\_write\_parallel\_data\_dual\_mode**

<b>Function name</b>	hpdf_write_parallel_data_dual_mode
<b>Function prototype</b>	void hpdf_write_parallel_data_dual_mode(hpdf_channel_enum channelx, int32_t data);
<b>Function descriptions</b>	write the parallel data on dual mode of data packing
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>data</b>	the parallel data
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* write the parallel data on dual mode of data packing */
```

```
hpdf_write_parallel_data_dual_mode(CHANNEL0, 0xEFFFFFFF);
```

### hpdf\_pulse\_skip\_update

The description of hpdf\_pulse\_skip\_update is shown as below:

**Table 3-584. Function hpdf\_pulse\_skip\_update**

Function name	hpdf_pulse_skip_update
Function prototype	void hpdf_pulse_skip_update(hpdf_channel_enum channelx, uint8_t number);
Function descriptions	update the number of pulses to skip
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module
CHANNELx(x=0..7)	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
Input parameter{in}	
number	the number of serial input samples that will be skipped
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* update the number of pulses to skip */
```

```
pdf_pulse_skip_update(CHANNEL0, 63);
```

### hpdf\_pulse\_skip\_read

The description of hpdf\_pulse\_skip\_read is shown as below:

**Table 3-585. Function hpdf\_pulse\_skip\_read**

Function name	hpdf_pulse_skip_read
Function prototype	uint8_t hpdf_pulse_skip_read(hpdf_channel_enum channelx);
Function descriptions	read the number of pulses to skip
Precondition	-
The called functions	-
Input parameter{in}	
channelx	The channel of HPDF module

<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint8_t</b>	the number of pulses to skip

Example:

```
/* read the number of pulses to skip */
uint8_t value;
value = hpdf_pulse_skip_read(CHANNEL0);
```

### hpdf\_filter\_enable

The description of hpdf\_filter\_enable is shown as below:

**Table 3-586. Function hpdf\_filter\_enable**

<b>Function name</b>	hpdf_filter_enable
<b>Function prototype</b>	void hpdf_filter_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable filter0 */
hpdf_filter_enable(FLT0);
```

### hpdf\_filter\_disable

The description of hpdf\_filter\_disable is shown as below:

**Table 3-587. Function hpdf\_filter\_disable**

<b>Function name</b>	hpdf_filter_disable
<b>Function prototype</b>	void hpdf_filter_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable filter0 */
```

```
hpdf_filter_disable(FLT0);
```

### hpdf\_filter\_config

The description of hpdf\_filter\_config is shown as below:

**Table 3-588. Function hpdf\_filter\_config**

<b>Function name</b>	hpdf_filter_config
<b>Function prototype</b>	void hpdf_filter_config(hpdf_filter_enum filtery, uint32_t order, uint16_t oversample);
<b>Function descriptions</b>	configure sinc filter order and oversample
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>order</b>	sinc filter order
<i>FLT_FASTSINC</i>	FastSinc filter type
<i>FLT_SINC1</i>	Sinc1 filter type
<i>FLT_SINC2</i>	Sinc2 filter type
<i>FLT_SINC3</i>	Sinc3 filter type
<i>FLT_SINC4</i>	Sinc4 filter type
<i>FLT_SINC5</i>	Sinc5 filter type
<b>Input parameter{in}</b>	
<b>oversample</b>	Sinc filter oversampling rate(1-1024)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure sinc filter order and oversample */
```

```
hpdf_filter_config(FLT0, FLT_SINC2, 64);
```



## hpdf\_integrator\_oversample

The description of hpdf\_integrator\_oversample is shown as below:

**Table 3-589. Function hpdf\_integrator\_oversample**

<b>Function name</b>	hpdf_integrator_oversample
<b>Function prototype</b>	void hpdf_integrator_oversample(hpdf_filter_enum filtery, uint8_t oversample);
<b>Function descriptions</b>	configure integrator oversampling rate
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>oversample</b>	integrator oversampling rate(1-256)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure integrator oversampling rate */
```

```
hpdf_integrator_oversample(FLT0, 256);
```

## hpdf\_threshold\_monitor\_filter\_config

The description of hpdf\_threshold\_monitor\_filter\_config is shown as below:

**Table 3-590. Function hpdf\_threshold\_monitor\_filter\_config**

<b>Function name</b>	hpdf_threshold_monitor_filter_config
<b>Function prototype</b>	void hpdf_threshold_monitor_filter_config(hpdf_channel_enum channelx, uint32_t order, uint8_t oversample);
<b>Function descriptions</b>	configure threshold monitor filter order and oversample
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Input parameter{in}</b>	
<b>order</b>	threshold monitor Sinc filter order
<i>TM_FASTSINC</i>	FastSinc filter type
<i>TM_SINC1</i>	Sinc1 filter type
<i>TM_SINC2</i>	Sinc2 filter type

<i>TM_SINC3</i>	Sinc3 filter type
<b>Input parameter{in}</b>	
<b>oversample</b>	Sinc filter oversampling rate(1-32)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure threshold monitor filter order and oversample */
```

```
hpdf_threshold_monitor_filter_config (FLT0, TM_SINC3);
```

### hpdf\_threshold\_monitor\_filter\_read\_data

The description of hpdf\_threshold\_monitor\_filter\_read\_data is shown as below:

**Table 3-591. Function hpdf\_threshold\_monitor\_filter\_read\_data**

<b>Function name</b>	hpdf_threshold_monitor_filter_read_data
<b>Function prototype</b>	int16_t hpdf_threshold_monitor_filter_read_data(hpdf_channel_enum channelx);
<b>Function descriptions</b>	read the threshold monitor filter data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int16_t</b>	the threshold monitor filter data

Example:

```
/* read the threshold monitor filter data */
```

```
int16_t data;
```

```
data = hpdf_threshold_monitor_filter_read_data(CHANNEL0);
```

### hpdf\_threshold\_monitor\_fast\_mode\_disable

The description of hpdf\_threshold\_monitor\_fast\_mode\_disable is shown as below:

**Table 3-592. Function hpdf\_threshold\_monitor\_fast\_mode\_disable**

<b>Function name</b>	hpdf_threshold_monitor_fast_mode_disable
<b>Function prototype</b>	void hpdf_threshold_monitor_fast_mode_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable threshold monitor fast mode

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable threshold monitor fast mode */
```

```
hpdf_threshold_monitor_fast_mode_disable (FLT0);
```

### hpdf\_threshold\_monitor\_fast\_mode\_enable

The description of hpdf\_threshold\_monitor\_fast\_mode\_enable is shown as below:

**Table 3-593. Function hpdf\_threshold\_monitor\_fast\_mode\_enable**

<b>Function name</b>	hpdf_threshold_monitor_fast_mode_enable
<b>Function prototype</b>	void hpdf_threshold_monitor_fast_mode_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable threshold monitor fast mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable threshold monitor fast mode */
```

```
hpdf_threshold_monitor_fast_mode_enable(FLT0);
```

### hpdf\_threshold\_monitor\_channel

The description of hpdf\_threshold\_monitor\_channel is shown as below:

**Table 3-594. Function hpdf\_threshold\_monitor\_channel**

<b>Function name</b>	hpdf_threshold_monitor_channel
<b>Function prototype</b>	void hpdf_threshold_monitor_channel(hpdf_filter_enum filtery, uint32_t channel);

<b>Function descriptions</b>	configure threshold monitor channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>channel</b>	which channel use threshold monitor x
<i>TMCHEN_DISABLE</i>	extremes monitor x does not accept data from any channel
<i>TMCHEN_CHANNELx</i>	extremes monitor accepts data from channel x
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure threshold monitor channel */
```

```
hpdf_threshold_monitor_channel(FLT0, TMCHEN_CHANNEL0_1);
```

### hpdf\_threshold\_monitor\_high\_threshold

The description of hpdf\_threshold\_monitor\_high\_threshold is shown as below:

**Table 3-595. Function hpdf\_threshold\_monitor\_high\_threshold**

<b>Function name</b>	hpdf_threshold_monitor_high_threshold
<b>Function prototype</b>	void hpdf_threshold_monitor_high_threshold(hpdf_filter_enum filtery, int32_t value);
<b>Function descriptions</b>	configure threshold monitor high threshold value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>value</b>	high threshold value(-8388608~8388607)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure threshold monitor high threshold value */
```

```
hpdf_threshold_monitor_high_threshold(FLT0, 32767);
```

## hpdf\_threshold\_monitor\_low\_threshold

The description of hpdf\_threshold\_monitor\_low\_threshold is shown as below:

**Table 3-596. Function hpdf\_threshold\_monitor\_low\_threshold**

<b>Function name</b>	hpdf_threshold_monitor_low_threshold
<b>Function prototype</b>	void hpdf_threshold_monitor_low_threshold(hpdf_filter_enum filtery, int32_t value);
<b>Function descriptions</b>	configure threshold monitor low threshold value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>value</b>	low threshold value(-8388608~8388607)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure threshold monitor low threshold value */
hpdf_threshold_monitor_low_threshold(FLT0, -32768);
```

## hpdf\_high\_threshold\_break\_signal

The description of hpdf\_high\_threshold\_break\_signal is shown as below:

**Table 3-597. Function hpdf\_high\_threshold\_break\_signal**

<b>Function name</b>	hpdf_high_threshold_break_signal
<b>Function prototype</b>	void hpdf_high_threshold_break_signal(hpdf_filter_enum filtery, uint32_t break_signal);
<b>Function descriptions</b>	configure threshold monitor high threshold event break signal
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>break_signal</b>	HPDF break signal
<i>NO_TM_HT_BREAK</i>	break signal is not distributed to an threshold monitor high threshold event
<i>TM_HT_BREAKx</i> (x=0..3)	break signal x is distributed to an threshold monitor high threshold event

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor high threshold event break signal */
hpdf_high_threshold_break_signal(FTL0, TM_HT_BREAK0);
```

### hpdf\_low\_threshold\_break\_signal

The description of hpdf\_low\_threshold\_break\_signal is shown as below:

**Table 3-598. Function hpdf\_low\_threshold\_break\_signal**

Function name	hpdf_low_threshold_break_signal
Function prototype	void hpdf_low_threshold_break_signal(hpdf_filter_enum filtery, uint32_t break_signal);
Function descriptions	configure threshold monitor low threshold event break signal
Precondition	-
The called functions	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
Input parameter{in}	
<b>break_signal</b>	HPDF break signal
<i>NO_TM_LT_BREAK</i>	break signal is not distributed to an threshold monitor low threshold event
<i>TM_LT_BREAKx</i> (x=0..3)	break signal x is distributed to an threshold monitor low threshold event
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure threshold monitor low threshold event break signal */
hpdf_low_threshold_break_signal(FTL0, TM_LT_BREAK0);
```

### hpdf\_extremes\_monitor\_channel

The description of hpdf\_extremes\_monitor\_channel is shown as below:

**Table 3-599. Function hpdf\_extremes\_monitor\_channel**

Function name	hpdf_extremes_monitor_channel
Function prototype	void hpdf_extremes_monitor_channel(hpdf_filter_enum filtery, uint32_t

	channel);
<b>Function descriptions</b>	configure extremes monitor channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>channel</b>	which channel use extremes monitor y (y=0..3)
<i>EM_CHANNEL_DISABLE</i>	extremes monitor y does not accept data from any channel
<i>EM_CHANNELx</i> (x=0..7)	extremes monitor accepts data from channel x
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure extremes monitor channel */
```

```
hpdf_extremes_monitor_channel(FTL0, EM_CHANNEL0);
```

### hpdf\_extremes\_monitor\_maximum\_get

The description of hpdf\_extremes\_monitor\_maximum\_get is shown as below:

**Table 3-600. Function hpdf\_extremes\_monitor\_maximum\_get**

<b>Function name</b>	hpdf_extremes_monitor_maximum_get
<b>Function prototype</b>	int32_t hpdf_extremes_monitor_maximum_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the extremes monitor maximum value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int32_t</b>	the maximum value

Example:

```
int32_t vlaue;
```

```
vlaue = hpdf_extremes_monitor_maximum_get(FTL0);
```

## hpdf\_extremes\_monitor\_minimum\_get

The description of hpdf\_extremes\_monitor\_minimum\_get is shown as below:

**Table 3-601. Function hpdf\_extremes\_monitor\_minimum\_get**

<b>Function name</b>	hpdf_extremes_monitor_minimum_get
<b>Function prototype</b>	int32_t hpdf_extremes_monitor_minimum_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the extremes monitor minimum value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int32_t</b>	the minimum value

Example:

```
/* get the extremes monitor minimum value */

int32_t vlaue;

vlaue = hpdf_extremes_monitor_minimum_get(FTL0,);
```

## hpdf\_conversion\_time\_get

The description of hpdf\_conversion\_time\_get is shown as below:

**Table 3-602. Function hpdf\_conversion\_time\_get**

<b>Function name</b>	hpdf_conversion_time_get
<b>Function prototype</b>	uint32_t hpdf_conversion_time_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the conversion timer value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int32_t</b>	the timer value

Example:

```
/* get the conversion timer value */

int32_t vlaue;
```



```
vlaue = hpdf_conversion_time_get(FTL0);
```

### hpdf\_rc\_continuous\_disable

The description of hpdf\_rc\_continuous\_disable is shown as below:

**Table 3-603. Function hpdf\_rc\_continuous\_disable**

<b>Function name</b>	hpdf_rc_continuous_disable
<b>Function prototype</b>	void hpdf_rc_continuous_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable regular conversions continuous mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable regular conversions continuous mode */
```

```
hpdf_rc_continuous_disable(FTL0);
```

### hpdf\_rc\_continuous\_enable

The description of hpdf\_rc\_continuous\_enable is shown as below:

**Table 3-604. Function hpdf\_rc\_continuous\_enable**

<b>Function name</b>	hpdf_rc_continuous_enable
<b>Function prototype</b>	void hpdf_rc_continuous_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable regular conversions continuous mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable regular conversions continuous mode */
```

```
hpdf_rc_continuous_enable(FTL0);
```

### hpdf\_rc\_start\_by\_software

The description of hpdf\_rc\_start\_by\_software is shown as below:

**Table 3-605. Function hpdf\_rc\_start\_by\_software**

<b>Function name</b>	hpdf_rc_start_by_software
<b>Function prototype</b>	void hpdf_rc_start_by_software(hpdf_filter_enum filtery);
<b>Function descriptions</b>	start regular channel conversion by software
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* start regular channel conversion by software */
```

```
hpdf_rc_start_by_software(FTL0);
```

### hpdf\_rc\_syn\_disable

The description of hpdf\_rc\_syn\_disable is shown as below:

**Table 3-606. Function hpdf\_rc\_syn\_disable**

<b>Function name</b>	hpdf_rc_syn_disable
<b>Function prototype</b>	void hpdf_rc_syn_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable regular conversion synchronously
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable regular conversion synchronously */
```

```
hpdf_rc_syn_disable(FTL0);
```

## hpdf\_rc\_syn\_enable

The description of hpdf\_rc\_syn\_enable is shown as below:

**Table 3-607. Function hpdf\_rc\_syn\_enable**

<b>Function name</b>	hpdf_rc_syn_enable
<b>Function prototype</b>	void hpdf_rc_syn_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable regular conversion synchronously
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable regular conversion synchronously */
```

```
hpdf_rc_syn_enable(FTL0);
```

## hpdf\_rc\_dma\_disable

The description of hpdf\_rc\_dma\_disable is shown as below:

**Table 3-608. Function hpdf\_rc\_dma\_disable**

<b>Function name</b>	hpdf_rc_dma_disable
<b>Function prototype</b>	void hpdf_rc_dma_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable regular conversion DMA channel
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable regular conversion DMA channel */
```

```
hpdf_rc_dma_disable(FTL0);
```

### hpdf\_rc\_dma\_enable

The description of hpdf\_rc\_dma\_enable is shown as below:

**Table 3-609. Function hpdf\_rc\_dma\_enable**

<b>Function name</b>	hpdf_rc_dma_enable
<b>Function prototype</b>	void hpdf_rc_dma_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable regular conversion DMA channel
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable regular conversion DMA channel */
```

```
hpdf_rc_dma_enable(FTL0);
```

### hpdf\_rc\_channel\_config

The description of hpdf\_rc\_channel\_config is shown as below:

**Table 3-610. Function hpdf\_rc\_channel\_config**

<b>Function name</b>	hpdf_rc_channel_config
<b>Function prototype</b>	void hpdf_rc_channel_config(hpdf_filter_enum filtery, hpdf_channel_enum channelx);
<b>Function descriptions</b>	configure regular conversion channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>channelx</b>	The channel of HPDF module
<i>CHANNELx(x=0..7)</i>	select HPDF channel, refer to <a href="#">Table 3-544. Enum hpdf_channel_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure regular conversion channel */
hpdf_rc_channel_config(FTL0, CHANNEL1);
```

### hpdf\_rc\_fast\_mode\_disable

The description of hpdf\_rc\_fast\_mode\_disable is shown as below:

**Table 3-611. Function hpdf\_rc\_fast\_mode\_disable**

<b>Function name</b>	hpdf_rc_fast_mode_disable
<b>Function prototype</b>	void hpdf_rc_fast_mode_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable regular conversion fast conversion mode
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable regular conversion fast conversion mode */
hpdf_rc_fast_mode_disable(FTL0);
```

### hpdf\_rc\_fast\_mode\_enable

The description of hpdf\_rc\_fast\_mode\_enable is shown as below:

**Table 3-612. Function hpdf\_rc\_fast\_mode\_enable**

<b>Function name</b>	hpdf_rc_fast_mode_enable
<b>Function prototype</b>	void hpdf_rc_fast_mode_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable regular conversion fast conversion mode
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable regular conversion fast conversion mode */
hpdf_rc_fast_mode_enable(FTL0);
```

## hpdf\_rc\_data\_get

The description of hpdf\_rc\_data\_get is shown as below:

**Table 3-613. Function hpdf\_rc\_data\_get**

<b>Function name</b>	hpdf_rc_data_get
<b>Function prototype</b>	int32_t hpdf_rc_data_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the regular conversion data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>int32_t</b>	regular conversions data

Example:

```
/* get the regular conversion data */
int32_t vlaue;
vlaue = hpdf_rc_data_get(FTL0);
```

## hpdf\_rc\_channel\_get

The description of hpdf\_rc\_channel\_get is shown as below:

**Table 3-614. Function hpdf\_rc\_channel\_get**

<b>Function name</b>	hpdf_rc_channel_get
<b>Function prototype</b>	uint8_t hpdf_rc_channel_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the channel of regular channel most recently converted
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

uint8_t	the channel
---------	-------------

Example:

```
/* get the channel of regular channel most recently converted */
```

```
uint8_t channel;
```

```
channel = hpdf_rc_channel_get(FTL0);
```

### hpdf\_ic\_start\_by\_software

The description of hpdf\_ic\_start\_by\_software is shown as below:

**Table 3-615. Function hpdf\_ic\_start\_by\_software**

<b>Function name</b>	hpdf_ic_start_by_software
<b>Function prototype</b>	void hpdf_ic_start_by_software(hpdf_filter_enum filtery);
<b>Function descriptions</b>	start inserted channel conversion by software
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* start inserted channel conversion by software */
```

```
hpdf_ic_start_by_software(FTL0);
```

### hpdf\_ic\_syn\_disable

The description of hpdf\_ic\_syn\_disable is shown as below:

**Table 3-616. Function hpdf\_ic\_syn\_disable**

<b>Function name</b>	hpdf_ic_syn_disable
<b>Function prototype</b>	void hpdf_ic_syn_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable inserted conversion synchronously
<b>Precondition</b>	disable FLTy(y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* disable inserted conversion synchronously */
```

```
hpdf_ic_syn_disable(FTL0);
```

### hpdf\_ic\_syn\_enable

The description of hpdf\_ic\_syn\_enable is shown as below:

**Table 3-617. Function hpdf\_ic\_syn\_enable**

Function name	hpdf_ic_syn_enable
Function prototype	void hpdf_ic_syn_enable(hpdf_filter_enum filtery);
Function descriptions	enable inserted conversion synchronously
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable inserted conversion synchronously */
```

```
hpdf_ic_syn_enable(FTL0);
```

### hpdf\_ic\_dma\_disable

The description of hpdf\_ic\_dma\_disable is shown as below:

**Table 3-618. Function hpdf\_ic\_dma\_disable**

Function name	hpdf_ic_dma_disable
Function prototype	void hpdf_ic_dma_disable(hpdf_filter_enum filtery);
Function descriptions	disable inserted conversion DMA channel
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-



Return value	
-	-

Example:

```
/* disable inserted conversion DMA channel */
```

```
hpdf_ic_dma_disable(FTL0);
```

### hpdf\_ic\_dma\_enable

The description of hpdf\_ic\_dma\_enable is shown as below:

**Table 3-619. Function hpdf\_ic\_dma\_enable**

<b>Function name</b>	hpdf_ic_dma_enable
<b>Function prototype</b>	void hpdf_ic_dma_enable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	enable inserted conversion DMA channel
<b>Precondition</b>	disable FLT <sub>y</sub> (y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
FLT <sub>y</sub> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable inserted conversion DMA channel */
```

```
hpdf_ic_dma_enable(FTL0);
```

### hpdf\_ic\_scan\_mode\_disable

The description of hpdf\_ic\_scan\_mode\_disable is shown as below:

**Table 3-620. Function hpdf\_ic\_scan\_mode\_disable**

<b>Function name</b>	hpdf_ic_scan_mode_disable
<b>Function prototype</b>	void hpdf_ic_scan_mode_disable(hpdf_filter_enum filtery);
<b>Function descriptions</b>	disable scan conversion mode
<b>Precondition</b>	disable FLT <sub>y</sub> (y=0..3)
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
FLT <sub>y</sub> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* disable scan conversion mode */
hpdf_ic_scan_mode_disable(FTL0);
```

### hpdf\_ic\_scan\_mode\_enable

The description of hpdf\_ic\_scan\_mode\_enable is shown as below:

**Table 3-621. Function hpdf\_ic\_scan\_mode\_enable**

Function name	hpdf_ic_scan_mode_enable
Function prototype	void hpdf_ic_scan_mode_enable(hpdf_filter_enum filtery);
Function descriptions	enable scan conversion mode
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable scan conversion mode */
hpdf_ic_scan_mode_enable(FTL0);
```

### hpdf\_ic\_trigger\_signal\_disable

The description of hpdf\_ic\_trigger\_signal\_disable is shown as below:

**Table 3-622. Function hpdf\_ic\_trigger\_signal\_disable**

Function name	hpdf_ic_trigger_signal_disable
Function prototype	void hpdf_ic_trigger_signal_disable(hpdf_filter_enum filtery);
Function descriptions	disable inserted conversions trigger signal
Precondition	disable FLTy(y=0..3)
The called functions	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* disable inserted conversions trigger signal */
```

```
hpdf_ic_trigger_signal_disable(FTL0);
```

## hpdf\_ic\_trigger\_signal\_config

The description of hpdf\_ic\_trigger\_signal\_config is shown as below:

**Table 3-623. Function hpdf\_ic\_trigger\_signal\_config**

Function name	hpdf_ic_trigger_signal_config
Function prototype	void hpdf_ic_trigger_signal_config(hpdf_filter_enum filtery, uint32_t trigger, uint32_t trigger_edge);
Function descriptions	configure inserted conversions trigger signal and trigger edge
Precondition	disable FLT <sub>y</sub> (y=0..3)
The called functions	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
FLT <sub>y</sub> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
Input parameter{in}	
<b>trigger</b>	inserted conversions trigger signal
HPDF_ITRG0	TIMER0_TRGO0 is selected to start inserted conversion
HPDF_ITRG1	TIMER0_TRGO1 is selected to start inserted conversion
HPDF_ITRG2	TIMER7_TRGO0 is selected to start inserted conversion
HPDF_ITRG3	TIMER7_TRGO1 is selected to start inserted conversion
HPDF_ITRG4	TIMER2_TRGO0 is selected to start inserted conversion
HPDF_ITRG5	TIMER3_TRGO0 is selected to start inserted conversion
HPDF_ITRG6	TIMER15_CH1 is selected to start inserted conversion
HPDF_ITRG7	TIMER5_TRGO0 is selected to start inserted conversion
HPDF_ITRG8	TIMER6_TRGO0 is selected to start inserted conversion
HPDF_ITRG11	TIMER22_TRGO0 is selected to start inserted conversion
HPDF_ITRG12	TIMER23_TRGO0 is selected to start inserted conversion
HPDF_ITRG24	EXTI11 is selected to start inserted conversion
HPDF_ITRG25	EXTI15 is selected to start inserted conversion
HPDF_ITRG31	HPDF_ITRG is selected to start inserted conversion
Input parameter{in}	
<b>trigger_edge</b>	inserted conversions trigger edge
TRG_DISABLE	disable trigger signal
RISING_EDGE_TRG	rising edge on the trigger signal
FALLING_EDGE_TRG	falling edge on the trigger signal
EDGE_TRG	edge (rising edges and falling edges) on the trigger signal
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* configure inserted conversions trigger signal and trigger edge */
```

```
hpdf_ic_trigger_signal_config(FTL0, HPDF_ITRG3, FALLING_EDGE_TRG);
```

### hpdf\_ic\_channel\_config

The description of hpdf\_ic\_channel\_config is shown as below:

**Table 3-624. Function hpdf\_ic\_channel\_config**

<b>Function name</b>	hpdf_ic_channel_config
<b>Function prototype</b>	void hpdf_ic_channel_config(hpdf_filter_enum filtery, uint32_t channel);
<b>Function descriptions</b>	configure inserted group conversions channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>channel</b>	the HPDF channel belongs to inserted group
<i>IGCSEL_CHANNELx</i> (x=0..7)	Channel x belongs to the inserted group
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure inserted group conversions channel */
```

```
hpdf_ic_channel_config(FTL0, IGCSEL_CHANNEL1);
```

### hpdf\_ic\_data\_get

The description of hpdf\_ic\_data\_get is shown as below:

**Table 3-625. Function hpdf\_ic\_data\_get**

<b>Function name</b>	hpdf_ic_data_get
<b>Function prototype</b>	int32_t hpdf_ic_data_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the inserted conversions data
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-
Return value	
<b>int32_t</b>	inserted conversions data

Example:

```
/* get the inserted conversions data */
int32_t vlaue;
vlaue = hpdf_ic_data_get(FTL0);
```

### hpdf\_ic\_channel\_get

The description of hpdf\_ic\_channel\_get is shown as below:

**Table 3-626. Function hpdf\_ic\_channel\_get**

<b>Function name</b>	hpdf_ic_channel_get
<b>Function prototype</b>	uint8_t hpdf_ic_channel_get(hpdf_filter_enum filtery);
<b>Function descriptions</b>	get the channel of inserted group channel most recently converted
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
Output parameter{out}	
-	-
Return value	
<b>uint8_t</b>	the channel

Example:

```
/* get the channel of inserted group channel most recently converted */
uint8_t channel;
channel = hpdf_ic_channel_get(FTL0);
```

### hpdf\_flag\_get

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

**Table 3-627. Function hpdf\_flag\_get**

<b>Function name</b>	hpdf_flag_get
<b>Function prototype</b>	FlagStatus hpdf_flag_get(hpdf_filter_enum filtery, hpdf_flag_enum flag);

<b>Function descriptions</b>	get the HPDF flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	HPDF flags, refer to <a href="#">Table 3-546. Enum hpdf_flag_enum</a>
<i>HPDF_FLAG_FLTy_IC EF</i>	FLTy inserted conversion end flag
<i>HPDF_FLAG_FLTy_RC EF</i>	FLTy regular conversion end flag
<i>HPDF_FLAG_FLTy_IC DOF</i>	FLTy inserted conversion data overflow flag
<i>HPDF_FLAG_FLTy_RC DOF</i>	FLTy threshold monitor event occurred flag
<i>HPDF_FLAG_FLTy_TM EOF</i>	FLTy inserted conversion in progress flag
<i>HPDF_FLAG_FLTy_IC PF</i>	FLTy regular conversion in progress flag
<i>HPDF_FLAG_FLTy_RC PF</i>	clock signal is lost on channel 0 flag
<i>HPDF_FLAG_FLT0_CK LFx</i>	clock signal is lost on channel x flag
<i>HPDF_FLAG_FLT0_M MFx</i>	malfunction event occurred on channel x flag
<i>HPDF_FLAG_FLTy_RC HPDT</i>	FLTy inserted channel most recently converted
<i>HPDF_FLAG_FLTy_LT Fx</i>	threshold monitor low threshold flag on channel x flag
<i>HPDF_FLAG_FLTy_HT Fx</i>	threshold monitor high threshold flag on channel x flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the FLT0 threshold monitor event occurred flag */
```

```
hpdf_flag_get(FTL0, HPDF_FLAG_FLTy_TMEOF);
```

## hpdf\_flag\_clear

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

**Table 3-628. Function hpdf\_flag\_clear**

<b>Function name</b>	hpdf_flag_clear
<b>Function prototype</b>	void hpdf_flag_clear(hpdf_filter_enum filtery, hpdf_flag_enum flag);
<b>Function descriptions</b>	clear the HPDF flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>flag</b>	HPDF flags, refer to <a href="#">Table 3-546. Enum hpdf_flag_enum</a>
<i>HPDF_FLAG_FLTy_ICEF</i>	FLTy inserted conversion end flag
<i>HPDF_FLAG_FLTy_RCEf</i>	FLTy regular conversion end flag
<i>HPDF_FLAG_FLTy_ICDOF</i>	FLTy inserted conversion data overflow flag
<i>HPDF_FLAG_FLTy_RCDOf</i>	FLTy threshold monitor event occurred flag
<i>HPDF_FLAG_FLTy_TMEOf</i>	FLTy inserted conversion in progress flag
<i>HPDF_FLAG_FLT0_CKLfx</i>	clock signal is lost on channel x flag
<i>HPDF_FLAG_FLT0_MMFx</i>	malfunction event occurred on channel x flag
<i>HPDF_FLAG_FLTy_LTFx</i>	threshold monitor low threshold flag on channel x flag
<i>HPDF_FLAG_FLTy_HTFx</i>	threshold monitor high threshold flag on channel x flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear the FLT0 threshold monitor event occurred flag */
```

```
hpdf_flag_clear(FTL0, HPDF_FLAG_FLTy_TMEOf);
```

## hpdf\_interrupt\_enable

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

**Table 3-629. Function hpdf\_interrupt\_enable**

<b>Function name</b>	hpdf_interrupt_enable
<b>Function prototype</b>	void hpdf_interrupt_enable(hpdf_filter_enum filtery, hpdf_interrput_enum interrupt);
<b>Function descriptions</b>	enable HPDF interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>interrupt</b>	HPDF interrupts, refer to <a href="#">Table 3-548. Enum hpdf_interrput_enum</a>
HPDF_INT_FLTy_ICEIE	FLTy inserted conversion end interrupt enable
HPDF_INT_FLTy_RCEIE	FLTy regular conversion end interrupt enable
HPDF_INT_FLTy_ICDOIE	FLTy inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_RCD OIE	FLTy regular conversion data overflow interrupt enable
HPDF_INT_FLTy_TMIE	FLTy threshold monitor interrupt enable
HPDF_INT_FLT0_MMI E	malfunction monitor interrupt enable
HPDF_INT_FLT0_CKLI E	clock loss interrupt enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FLT0 threshold monitor interrupt */
```

```
hpdf_interrupt_enable(FTL0, HPDF_INT_FLTy_TMEIOIE);
```

## hpdf\_interrupt\_disable

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

**Table 3-630. Function hpdf\_interrupt\_disable**

<b>Function name</b>	hpdf_interrupt_disable
<b>Function prototype</b>	void hpdf_interrupt_disable(hpdf_filter_enum filtery, hpdf_interrput_enum



	interrupt);
<b>Function descriptions</b>	disable HPDF interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy(y=0..3)</i>	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>interrupt</b>	HPDF interrupts, refer to <a href="#">Table 3-548. Enum hpdf_interrupt_enum</a>
HPDF_INT_FLTy_ICEIE	FLTy inserted conversion end interrupt enable
HPDF_INT_FLTy_RCEIE	FLTy regular conversion end interrupt enable
HPDF_INT_FLTy_ICDOIE	FLTy inserted conversion data overflow interrupt enable
HPDF_INT_FLTy_RCDOIE	FLTy regular conversion data overflow interrupt enable
HPDF_INT_FLTy_TMIE	FLTy threshold monitor interrupt enable
HPDF_INT_FLT0_MMIE	malfunction monitor interrupt enable
HPDF_INT_FLT0_CKLEIE	clock loss interrupt enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FLT0 threshold monitor interrupt */
```

```
hpdf_interrupt_disable(FLT0, HPDF_INT_FLT0_CKLEIE);
```

### hpdf\_interrupt\_flag\_get

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

**Table 3-631. Function hpdf\_interrupt\_flag\_get**

<b>Function name</b>	hpdf_interrupt_flag_get
<b>Function prototype</b>	FlagStatus hpdf_interrupt_flag_get(hpdf_filter_enum filtery, hpdf_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	get the HPDF interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module

<i>FLTy</i> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>int_flag</b>	HPDF flags, refer to <a href="#">Table 3-547. Enum hpdf_interrupt_flag_enum</a>
HPDF_INT_FLAG_FLT y_ICEF	FLTy inserted conversion end interrupt flag
HPDF_INT_FLAG_FLT y_RCEF	FLTy regular conversion end interrupt flag
HPDF_INT_FLAG_FLT y_ICDOF	FLTy inserted conversion data overflow interrupt flag
HPDF_INT_FLAG_FLT y_RCDOF	FLTy regular conversion data overflow interrupt flag
HPDF_INT_FLAG_FLT y_TMEOF	FLTy threshold monitor event occurred interrupt flag
HPDF_INT_FLAG_FLT 0_CKLFx	clock signal is lost on channel x interrupt flag
HPDF_INT_FLAG_FLT 0_MMFx	malfunction event occurred on channel x interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the the clock loss interrupt flag */
```

```
hpdf_interrupt_flag_get (FTL0, HPDF_INT_FLAG_FLT0_CKLF0);
```

### hpdf\_interrupt\_flag\_clear

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

**Table 3-632. Function hpdf\_interrupt\_flag\_clear**

<b>Function name</b>	hpdf_interrupt_flag_clear
<b>Function prototype</b>	void hpdf_interrupt_flag_clear(hpdf_filter_enum filtery, hpdf_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	clear the HPDF interrupt flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>filtery</b>	The filter of HPDF module
<i>FLTy</i> (y=0..3)	select HPDF filter, refer to <a href="#">Table 3-545. Enum hpdf_filter_enum</a>
<b>Input parameter{in}</b>	
<b>int_flag</b>	HPDF flags, refer to <a href="#">Table 3-547. Enum hpdf_interrupt_flag_enum</a>
HPDF_INT_FLAG_FLT y_ICEF	FLTy inserted conversion end interrupt flag

HPDF_INT_FLAG_FLT y_RCEF	FLTy regular conversion end interrupt flag
HPDF_INT_FLAG_FLT y_ICDOF	FLTy inserted conversion data overflow interrupt flag
HPDF_INT_FLAG_FLT y_RCDOF	FLTy regular conversion data overflow interrupt flag
HPDF_INT_FLAG_FLT y_TMEOF	FLTy threshold monitor event occurred interrupt flag
HPDF_INT_FLAG_FLT 0_CKLFx	clock signal is lost on channelx interrupt flag
HPDF_INT_FLAG_FLT 0_MMFX	malfunction event occurred on channelx interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example :

```
/* clear the the clock loss interrupt flag */
```

```
hpdf_interrupt_flag_clear(FTL0, HPDF_INT_FLAG_FLT0_CKLF0);
```

## 3.20. I2C

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. The I2C registers are listed in chapter [3.20.1](#), the I2C firmware functions are introduced in chapter [3.20.2](#).

### 3.20.1. Descriptions of Peripheral registers

I2C registers are listed in the table shown as below:

**Table 3-633. 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

Registers	Descriptions
I2C_RDATA	I2C receive data register
I2C_TDATA	I2C transmit data register
I2C_CTL2	I2C control register 2

### 3.20.2. Descriptions of Peripheral functions

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

**Table 3-634. I2C firmware function**

Function name	Function description
i2c_deinit	reset I2C
i2c_timing_config	configure the timing parameters
i2c_digital_noise_filter_config	configure digital noise filter
i2c_analog_noise_filter_enable	enable analog noise filter
i2c_analog_noise_filter_disable	disable analog noise filter
i2c_master_clock_config	configure the SCL high and low period of clock in master mode
i2c_master_addressing	configure I2C slave address and transfer direction in master mode
i2c_address10_header_enable	10-bit address header executes read direction only in master receive mode
i2c_address10_header_disable	10-bit address header executes complete sequence in master receive mode
i2c_address10_enable	enable 10-bit addressing mode in master mode
i2c_address10_disable	disable 10-bit addressing mode in master mode
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_receivied_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

Function name	Function description
i2c_nack_disable	generate an ACK in slave mode
i2c_wakeup_from_deepsleep_enable	enable wakeup from deep-sleep mode
i2c_wakeup_from_deepsleep_disable	disable wakeup from deep-sleep mode
i2c_enable	enable I2C
i2c_disable	disable I2C
i2c_start_on_bus	generate a START condition on I2C bus
i2c_stop_on_bus	generate a STOP condition on I2C bus
i2c_data_transmit	I2C transmit data
i2c_data_receive	I2C receive data
i2c_reload_enable	enable I2C reload mode
i2c_reload_disable	disable I2C reload mode
i2c_transfer_byte_number_config	configure number of bytes to be transferred
i2c_dma_enable	enable I2C DMA for transmission or reception
i2c_dma_disable	disable I2C DMA for transmission or reception
i2c_pec_transfer	I2C transfers PEC value
i2c_pec_enable	enable I2C PEC calculation
i2c_pec_disable	disable I2C PEC calculation
i2c_pec_value_get	get packet error checking value
i2c_smbus_alert_enable	enable SMBus alert
i2c_smbus_alert_disable	disable SMBus alert
i2c_smbus_default_addr_enable	enable SMBus device default address
i2c_smbus_default_addr_disable	disable SMBus device default address
i2c_smbus_host_addr_enable	enable SMBus host address
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 3-635. 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 3-636. Function i2c\_deinit**

<b>Function name</b>	i2c_deinit
<b>Function prototype</b>	void i2c_deinit(uint32_t i2c_periph);
<b>Function descriptions</b>	reset I2C
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset I2C0 */
i2c_deinit(I2C0);
```

## i2c\_timing\_config

The description of i2c\_timing\_config is shown as below:

Table 3-637. Function i2c\_timing\_config

Function name	i2c_timing_config
Function prototype	void i2c_timing_config(uint32_t i2c_periph, uint32_t psc, uint32_t scl_dely, uint32_t sda_dely);
Function descriptions	configure the timing parameters
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
psc	0-0xf, timing prescaler
Input parameter{in}	
scl_dely	0-0xf, data setup time
Input parameter{in}	
sda_dely	0-0xf, data hold time
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the timing parameters */
i2c_timing_config(I2C0, 0x1, 0x2, 0x1);
```

### i2c\_digital\_noise\_filter\_config

The description of i2c\_digital\_noise\_filter\_config is shown as below:

Table 3-638. Function i2c\_digital\_noise\_filter\_config

Function name	i2c_digital_noise_filter_config
Function prototype	void i2c_digital_noise_filter_config(uint32_t i2c_periph, uint32_t filter_length);
Function descriptions	configure digital noise filter
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
filter_length	filter_length
FILTER_DISABLE	digital filter is disabled
FILTER_LENGTH_1	digital filter is enabled and filter spikes with a length of up to 1 t <sub>I2CCLK</sub>
FILTER_LENGTH_2	digital filter is enabled and filter spikes with a length of up to 2 t <sub>I2CCLK</sub>

<i>FILTER_LENGTH_3</i>	digital filter is enabled and filter spikes with a length of up to 3 $t_{I2CCLK}$
<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:

```
/* I2C0 digital filter filters spikes with a length of up to 1  $t_{I2CCLK}$  */
```

```
i2c_digital_noise_filter_config(I2C0, FILTER_LENGTH_1);
```

### i2c\_analog\_noise\_filter\_enable

The description of i2c\_analog\_noise\_filter\_enable is shown as below:

**Table 3-639. Function i2c\_analog\_noise\_filter\_enable**

<b>Function name</b>	i2c_analog_noise_filter_enable
<b>Function prototype</b>	void i2c_analog_noise_filter_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable analog noise filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable analog noise filter */
```

```
i2c_analog_noise_filter_enable(I2C0);
```



## i2c\_analog\_noise\_filter\_disable

The description of i2c\_analog\_noise\_filter\_disable is shown as below:

**Table 3-640. Function i2c\_analog\_noise\_filter\_disable**

<b>Function name</b>	i2c_analog_noise_filter_disable
<b>Function prototype</b>	void i2c_analog_noise_filter_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable analog noise filter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable analog noise filter */
```

```
i2c_analog_noise_filter_disable(I2C0);
```

## i2c\_master\_clock\_config

The description of i2c\_master\_clock\_config is shown as below:

**Table 3-641. Function i2c\_master\_clock\_config**

<b>Function name</b>	i2c_master_clock_config
<b>Function prototype</b>	void i2c_master_clock_config(uint32_t i2c_periph, uint32_t sclh, uint32_t scll);
<b>Function descriptions</b>	configure the SCL high and low period of clock in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>sclh</b>	0-0xff, SCL high period
<b>Input parameter{in}</b>	
<b>scll</b>	0-0xff, SCL low period
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the SCL and SDA period of clock in master mode */
i2c_master_clock_config(I2C0, 0x0f, 0x0f);
```

### i2c\_master\_addressing

The description of i2c\_master\_addressing is shown as below:

**Table 3-642. Function i2c\_master\_addressing**

<b>Function name</b>	i2c_master_addressing
<b>Function prototype</b>	void i2c_master_addressing(uint32_t i2c_periph, uint32_t address, uint32_t trans_direction);
<b>Function descriptions</b>	configure i2c slave addresss and transfer direction in master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>address</b>	0-0x3FF except reserved address, I2C slave address to be sent
<b>Input parameter{in}</b>	
<b>trans_direction</b>	I2C transfer direction in master mode
<i>I2C_MASTER_TRANS MIT</i>	master transmit
<i>I2C_MASTER_RECEIV E</i>	master receive
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* send slave address to I2C bus */
i2c_master_addressing(I2C0, 0x82, I2C_MASTER_TRANSMIT);
```

### i2c\_address10\_header\_enable

The description of i2c\_address10\_header\_enable is shown as below:

**Table 3-643. Function i2c\_address10\_header\_enable**

<b>Function name</b>	i2c_address10_header_enable
<b>Function prototype</b>	void i2c_address10_header_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	10-bit address header executes read direction only in master receive mode
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* 10-bit address header executes read direction only in master receive mode */
```

```
i2c_address10_header_enable(I2C0);
```

### i2c\_address10\_header\_disable

The description of i2c\_address10\_header\_disable is shown as below:

**Table 3-644. Function i2c\_address10\_header\_disable**

Function name	i2c_address10_header_disable
Function prototype	void i2c_address10_header_disable(uint32_t i2c_periph);
Function descriptions	10-bit address header executes complete sequence in master receive mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* 10-bit address header executes complete sequence in master receive mode */
```

```
i2c_address10_header_disable(I2C0);
```

### i2c\_address10\_enable

The description of i2c\_address10\_enable is shown as below:

**Table 3-645. Function i2c\_address10\_enable**

Function name	i2c_address10_enable
Function prototype	void i2c_address10_enable(uint32_t i2c_periph);
Function descriptions	enable 10-bit addressing mode in master mode
Precondition	-

The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable 10-bit addressing mode in master mode */
```

```
i2c_address10_enable(I2C0);
```

### i2c\_address10\_disable

The description of i2c\_address10\_disable is shown as below:

**Table 3-646. Function i2c\_address10\_disable**

Function name	i2c_address10_disable
Function prototype	void i2c_address10_disable(uint32_t i2c_periph);
Function descriptions	disable 10-bit addressing mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable 10-bit addressing mode in master mode */
```

```
i2c_address10_disable(I2C0);
```

### i2c\_automatic\_end\_enable

The description of i2c\_automatic\_end\_enable is shown as below:

**Table 3-647. Function i2c\_automatic\_end\_enable**

Function name	i2c_automatic_end_enable
Function prototype	void i2c_automatic_end_enable(uint32_t i2c_periph);
Function descriptions	enable I2C automatic end mode in master mode
Precondition	-

The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C automatic end mode in master mode */
```

```
i2c_automatic_end_enable(I2C0);
```

### i2c\_automatic\_end\_disable

The description of i2c\_automatic\_end\_disable is shown as below:

**Table 3-648. Function i2c\_automatic\_end\_disable**

Function name	i2c_automatic_end_disable
Function prototype	void i2c_automatic_end_disable(uint32_t i2c_periph);
Function descriptions	disable I2C automatic end mode in master mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C automatic end mode in master mode */
```

```
i2c_automatic_end_disable(I2C0);
```

### i2c\_slave\_response\_to\_gcall\_enable

The description of i2c\_slave\_response\_to\_gcall\_enable is shown as below:

**Table 3-649. Function i2c\_slave\_response\_to\_gcall\_enable**

Function name	i2c_slave_response_to_gcall_enable
Function prototype	void i2c_slave_response_to_gcall_enable(uint32_t i2c_periph);
Function descriptions	enable the response to a general call
Precondition	-

The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the response to a general call */
```

```
i2c_slave_response_to_gcall_enable(I2C0);
```

### i2c\_slave\_response\_to\_gcall\_disable

The description of i2c\_slave\_response\_to\_gcall\_disable is shown as below:

**Table 3-650. Function i2c\_slave\_response\_to\_gcall\_disable**

Function name	i2c_slave_response_to_gcall_disable
Function prototype	void i2c_slave_response_to_gcall_disable(uint32_t i2c_periph);
Function descriptions	disable the response to a general call
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the response to a general call */
```

```
i2c_slave_response_to_gcall_disable(I2C0);
```

### i2c\_stretch\_scl\_low\_enable

The description of i2c\_stretch\_scl\_low\_enable is shown as below:

**Table 3-651. Function i2c\_stretch\_scl\_low\_enable**

Function name	i2c_stretch_scl_low_enable
Function prototype	void i2c_stretch_scl_low_enable(uint32_t i2c_periph);
Function descriptions	enable to stretch SCL low when data is not ready in slave mode
Precondition	-

The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
i2c_stretch_scl_low_enable(I2C0);
```

### i2c\_stretch\_scl\_low\_disable

The description of i2c\_stretch\_scl\_low\_disable is shown as below:

**Table 3-652. Function i2c\_stretch\_scl\_low\_disable**

Function name	i2c_stretch_scl_low_disable
Function prototype	void i2c_stretch_scl_low_disable(uint32_t i2c_periph);
Function descriptions	disable to stretch SCL low when data is not ready in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

```
i2c_stretch_scl_low_disable(I2C0);
```

### i2c\_address\_config

The description of i2c\_address\_config is shown as below:

**Table 3-653. Function i2c\_address\_config**

Function name	i2c_address_config
Function prototype	void i2c_address_config(uint32_t i2c_periph, uint32_t address, uint32_t addr_format);
Function descriptions	configure i2c slave address

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>address</b>	I2C address
<b>Input parameter{in}</b>	
<b>addr_format</b>	7 bits or 10 bits
<i>I2C_ADDFORMAT_7BITS</i>	7bits
<i>I2C_ADDFORMAT_10BITS</i>	10bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure i2c slave address */
```

```
i2c_address_config(I2C0, 0x82, I2C_ADDFORMAT_7BITS);
```

### i2c\_address\_bit\_compare\_config

The description of i2c\_address\_bit\_compare\_config is shown as below:

**Table 3-654. 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 i2c_periph, 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>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>compare_bits</b>	the bits need to compare
<i>ADDRESS_BIT1_COMPARE</i>	address bit1 needs compare
<i>ADDRESS_BIT2_COMPARE</i>	address bit2 needs compare
<i>ADDRESS_BIT3_COMPARE</i>	address bit3 needs compare



<i>PARE</i>	
<i>ADDRESS_BIT4_COM</i> <i>PARE</i>	address bit4 needs compare
<i>ADDRESS_BIT5_COM</i> <i>PARE</i>	address bit5 needs compare
<i>ADDRESS_BIT6_COM</i> <i>PARE</i>	address bit6 needs compare
<i>ADDRESS_BIT7_COM</i> <i>PARE</i>	address bit7 needs compare
<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(I2C0, ADDRESS_BIT1_COMPARE);
```

### i2c\_address\_disable

The description of i2c\_address\_disable is shown as below:

**Table 3-655. Function i2c\_address\_disable**

<b>Function name</b>	i2c_address_disable
<b>Function prototype</b>	void i2c_address_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable i2c address in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable i2c address in slave mode */
i2c_address_disable(I2C0);
```

### i2c\_second\_address\_config

The description of i2c\_second\_address\_config is shown as below:

Table 3-656. Function i2c\_second\_address\_config

Function name	i2c_second_address_config
Function prototype	void i2c_second_address_config(uint32_t i2c_periph, uint32_t address, uint32_t addr_mask);
Function descriptions	configure i2c second slave address
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
address	I2C address
Input parameter{in}	
addr_mask	the bits not need to compare
ADDRESS2_NO_MASK	no mask, all the bits must be compared
ADDRESS2_MASK_BIT1	ADDRESS2[1] is masked, only ADDRESS2[7:2] are compared
ADDRESS2_MASK_BIT1_2	ADDRESS2[2:1] is masked, only ADDRESS2[7:3] are compared
ADDRESS2_MASK_BIT1_3	ADDRESS2[3:1] is masked, only ADDRESS2[7:4] are compared
ADDRESS2_MASK_BIT1_4	ADDRESS2[4:1] is masked, only ADDRESS2[7:5] are compared
ADDRESS2_MASK_BIT1_5	ADDRESS2[5:1] is masked, only ADDRESS2[7:6] are compared
ADDRESS2_MASK_BIT1_6	ADDRESS2[6:1] is masked, only ADDRESS2[7] are compared
ADDRESS2_MASK_ALL	all the ADDRESS2[7:1] bits are masked
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure i2c second slave address */
```

```
i2c_second_address_config(I2C0, 0x82, ADDRESS2_MASK_BIT1_2);
```

### i2c\_second\_address\_disable

The description of i2c\_second\_address\_disable is shown as below:

Table 3-657. Function i2c\_second\_address\_disable

Function name	i2c_second_address_disable
Function prototype	void i2c_second_address_disable(uint32_t i2c_periph);
Function descriptions	disable i2c second address in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable i2c second address in slave mode */
```

```
i2c_second_address_disable(I2C0);
```

### i2c\_receved\_address\_get

The description of i2c\_receved\_address\_get is shown as below:

Table 3-658. Function i2c\_receved\_address\_get

Function name	i2c_receved_address_get
Function prototype	uint32_t i2c_receved_address_get(uint32_t i2c_periph);
Function descriptions	get received match address in slave mode
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Output parameter{out}	
-	-
Return value	
uint32_t	0x00..0x7F

Example:

```
/* get received match address in slave mode */
```

```
uint32_t address;
```

```
address = i2c_receved_address_get(I2C0);
```

## i2c\_slave\_byte\_control\_enable

The description of i2c\_slave\_byte\_control\_enable is shown as below:

**Table 3-659. Function i2c\_slave\_byte\_control\_enable**

<b>Function name</b>	i2c_slave_byte_control_enable
<b>Function prototype</b>	void i2c_slave_byte_control_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable slave byte control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable slave byte control */

i2c_slave_byte_control_enable(I2C0);
```

## i2c\_slave\_byte\_control\_disable

The description of i2c\_slave\_byte\_control\_disable is shown as below:

**Table 3-660. Function i2c\_slave\_byte\_control\_disable**

<b>Function name</b>	i2c_slave_byte_control_disable
<b>Function prototype</b>	void i2c_slave_byte_control_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable slave byte control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable slave byte control */

i2c_slave_byte_control_disable(I2C0);
```

## i2c\_nack\_enable

The description of i2c\_nack\_enable is shown as below:

**Table 3-661. Function i2c\_nack\_enable**

<b>Function name</b>	i2c_nack_enable
<b>Function prototype</b>	void i2c_nack_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	generate a NACK in slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* generate a NACK in slave mode */
```

```
i2c_nack_enable(I2C0);
```

## i2c\_wakeup\_from\_deepsleep\_enable

The description of i2c\_wakeup\_from\_deepsleep\_enable is shown as below:

**Table 3-662. Function i2c\_wakeup\_from\_deepsleep\_enable**

<b>Function name</b>	i2c_wakeup_from_deepsleep_enable
<b>Function prototype</b>	void i2c_wakeup_from_deepsleep_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable wakeup from Deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable wakeup from Deep-sleep mode */
```

```
i2c_wakeup_from_deepsleep_enable(I2C0);
```

## i2c\_wakeup\_from\_deepsleep\_disable

The description of i2c\_wakeup\_from\_deepsleep\_disable is shown as below:

**Table 3-663. Function i2c\_wakeup\_from\_deepsleep\_disable**

<b>Function name</b>	i2c_wakeup_from_deepsleep_disable
<b>Function prototype</b>	void i2c_wakeup_from_deepsleep_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable wakeup from Deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable wakeup from Deep-sleep mode */
i2c_wakeup_from_deepsleep_disable(I2C0);
```

## i2c\_enable

The description of i2c\_enable is shown as below:

**Table 3-664. Function i2c\_enable**

<b>Function name</b>	i2c_enable
<b>Function prototype</b>	void i2c_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable I2C
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I2C0 */
i2c_enable(I2C0);
```

## i2c\_disable

The description of i2c\_disable is shown as below:

**Table 3-665. Function i2c\_disable**

<b>Function name</b>	i2c_disable
<b>Function prototype</b>	void i2c_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable I2C
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable I2C0 */
i2c_disable(I2C0);
```

## i2c\_start\_on\_bus

The description of i2c\_start\_on\_bus is shown as below:

**Table 3-666. Function i2c\_start\_on\_bus**

<b>Function name</b>	i2c_start_on_bus
<b>Function prototype</b>	void i2c_start_on_bus(uint32_t i2c_periph);
<b>Function descriptions</b>	generate a START condition on I2C bus
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* I2C0 send a start condition to I2C bus */
i2c_start_on_bus(I2C0);
```

## i2c\_stop\_on\_bus

The description of i2c\_stop\_on\_bus is shown as below:

**Table 3-667. Function i2c\_stop\_on\_bus**

<b>Function name</b>	i2c_stop_on_bus
<b>Function prototype</b>	void i2c_stop_on_bus(uint32_t i2c_periph);
<b>Function descriptions</b>	generate a STOP condition on I2C bus
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* I2C0 generate a STOP condition to I2C bus */
```

```
i2c_stop_on_bus(I2C0);
```

## i2c\_data\_transmit

The description of i2c\_data\_transmit is shown as below:

**Table 3-668. Function i2c\_data\_transmit**

<b>Function name</b>	i2c_data_transmit
<b>Function prototype</b>	void i2c_data_transmit(uint32_t i2c_periph, uint32_t data);
<b>Function descriptions</b>	I2C transmit data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>data</b>	transmit data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* I2C0 transmit data */
```



```
i2c_data_transmit(I2C0, 0x80);
```

## i2c\_data\_receive

The description of i2c\_data\_receive is shown as below:

**Table 3-669. Function i2c\_data\_receive**

<b>Function name</b>	i2c_data_receive
<b>Function prototype</b>	uint32_t i2c_data_receive(uint32_t i2c_periph);
<b>Function descriptions</b>	I2C receive data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	0x0000..0x00FF

Example:

```
/* I2C0 receive data */
```

```
uint32_t i2c_receiver;
```

```
i2c_receiver = i2c_data_receive(I2C0);
```

## i2c\_reload\_enable

The description of i2c\_reload\_enable is shown as below:

**Table 3-670. Function i2c\_reload\_enable**

<b>Function name</b>	i2c_reload_enable
<b>Function prototype</b>	void i2c_reload_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable I2C reload mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I2C reload mode */
```

```
i2c_reload_enable(I2C0);
```

### i2c\_reload\_disable

The description of i2c\_reload\_disable is shown as below:

**Table 3-671. Function i2c\_reload\_disable**

<b>Function name</b>	i2c_reload_disable
<b>Function prototype</b>	void i2c_reload_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable I2C reload mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable I2C reload mode */
```

```
i2c_reload_disable(I2C0);
```

### i2c\_transfer\_byte\_number\_config

The description of i2c\_transfer\_byte\_number\_config is shown as below:

**Table 3-672. 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(uint32_t i2c_periph, uint32_t byte_number);
<b>Function descriptions</b>	configure number of bytes to be transferred
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>byte_number</b>	0x0-0xFF, number of bytes to be transferred
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* configure number of bytes to be transferred */
```

```
i2c_transfer_byte_number_config(I2C0, 0xFF);
```

### i2c\_dma\_enable

The description of i2c\_dma\_enable is shown as below:

**Table 3-673. Function i2c\_dma\_enable**

Function name	i2c_dma_enable
Function prototype	void i2c_dma_enable(uint32_t i2c_periph, uint8_t dma);
Function descriptions	enable I2C DMA for transmission or reception
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
dma	I2C DMA
I2C_DMA_TRANSMIT	transmit data using DMA
I2C_DMA_RECEIVE	receive data using DMA
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C DMA for transmission or reception */
```

```
i2c_dma_enable(I2C0, I2C_DMA_RECEIVE);
```

### i2c\_dma\_disable

The description of i2c\_dma\_disable is shown as below:

**Table 3-674. Function i2c\_dma\_disable**

Function name	i2c_dma_disable
Function prototype	void i2c_dma_disable(uint32_t i2c_periph, uint8_t dma);
Function descriptions	disable I2C DMA for transmission or reception
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral

<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>dma</b>	I2C DMA
<i>I2C_DMA_TRANSMIT</i>	transmit data using DMA
<i>I2C_DMA_RECEIVE</i>	receive data using DMA
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable I2C DMA for transmission or reception */
```

```
i2c_dma_disable(I2C0, I2C_DMA_RECEIVE);
```

### i2c\_pec\_transfer

The description of i2c\_pec\_transfer is shown as below:

**Table 3-675. Function i2c\_pec\_transfer**

<b>Function name</b>	i2c_pec_transfer
<b>Function prototype</b>	void i2c_pec_transfer(uint32_t i2c_periph);
<b>Function descriptions</b>	I2C transfers PEC value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* I2C transfers PEC value */
```

```
i2c_pec_transfer(I2C0);
```

### i2c\_pec\_enable

The description of i2c\_pec\_enable is shown as below:

**Table 3-676. Function i2c\_pec\_enable**

<b>Function name</b>	i2c_pec_enable
<b>Function prototype</b>	void i2c_pec_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable I2C PEC calculation

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I2C PEC calculation */
```

```
i2c_pec_enable(I2C0);
```

### i2c\_pec\_disable

The description of i2c\_pec\_disable is shown as below:

**Table 3-677. Function i2c\_pec\_disable**

<b>Function name</b>	i2c_pec_disable
<b>Function prototype</b>	void i2c_pec_disable(uint32_t i2c_periph);
<b>Function descriptions</b>	disable I2C PEC calculation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable I2C PEC calculation */
```

```
i2c_pec_disable(I2C0);
```

### i2c\_pec\_value\_get

The description of i2c\_pec\_value\_get is shown as below:

**Table 3-678. Function i2c\_pec\_value\_get**

<b>Function name</b>	i2c_pec_value_get
<b>Function prototype</b>	uint32_t i2c_pec_value_get(uint32_t i2c_periph);
<b>Function descriptions</b>	get packet error checking value

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	PEC value

Example:

```
/* I2C0 get packet error checking value */
```

```
uint32_t pec_value;
```

```
pec_value = i2c_pec_value_get(I2C0);
```

### i2c\_smbus\_alert\_enable

The description of i2c\_smbus\_alert\_enable is shown as below:

**Table 3-679. Function i2c\_smbus\_alert\_enable**

<b>Function name</b>	i2c_smbus_alert_enable
<b>Function prototype</b>	void i2c_smbus_alert_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable SMBus Alert
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SMBus Alert */
```

```
i2c_smbus_alert_enable(I2C0);
```

### i2c\_smbus\_alert\_disable

The description of i2c\_smbus\_alert\_disable is shown as below:

**Table 3-680. Function i2c\_smbus\_alert\_disable**

<b>Function name</b>	i2c_smbus_alert_disable
<b>Function prototype</b>	void i2c_smbus_alert_disable(uint32_t i2c_periph);

<b>Function descriptions</b>	disable SMBus Alert
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SMBus Alert */
```

```
i2c_smbus_alert_disable(I2C0);
```

### **i2c\_smbus\_default\_addr\_enable**

The description of i2c\_smbus\_default\_addr\_enable is shown as below:

**Table 3-681. Function i2c\_smbus\_default\_addr\_enable**

<b>Function name</b>	i2c_smbus_default_addr_enable
<b>Function prototype</b>	void i2c_smbus_default_addr_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable SMBus device default address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SMBus device default address */
```

```
i2c_smbus_default_addr_enable(I2C0);
```

### **i2c\_smbus\_default\_addr\_disable**

The description of i2c\_smbus\_default\_addr\_disable is shown as below:

**Table 3-682. Function i2c\_smbus\_default\_addr\_disable**

<b>Function name</b>	i2c_smbus_default_addr_disable
<b>Function prototype</b>	void i2c_smbus_default_addr_disable(uint32_t i2c_periph);

<b>Function descriptions</b>	disable SMBus device default address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SMBus device default address */
i2c_smbus_default_addr_disable(I2C0);
```

### i2c\_smbus\_host\_addr\_enable

The description of i2c\_smbus\_host\_addr\_enable is shown as below:

**Table 3-683. Function i2c\_smbus\_host\_addr\_enable**

<b>Function name</b>	i2c_smbus_host_addr_enable
<b>Function prototype</b>	void i2c_smbus_host_addr_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable SMBus Host address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SMBus Host address */
i2c_smbus_host_addr_enable(I2C0);
```

### i2c\_smbus\_host\_addr\_disable

The description of i2c\_smbus\_host\_addr\_disable is shown as below:

**Table 3-684. Function i2c\_smbus\_host\_addr\_disable**

<b>Function name</b>	i2c_smbus_host_addr_disable
<b>Function prototype</b>	void i2c_smbus_host_addr_disable(uint32_t i2c_periph);



<b>Function descriptions</b>	disable SMBus Host address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SMBus Host address */
```

```
i2c_smbus_host_addr_disable(I2C0);
```

### i2c\_extented\_clock\_timeout\_enable

The description of i2c\_extented\_clock\_timeout\_enable is shown as below:

**Table 3-685. Function i2c\_extented\_clock\_timeout\_enable**

<b>Function name</b>	i2c_extented_clock_timeout_enable
<b>Function prototype</b>	void i2c_extented_clock_timeout_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable extended clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable extended clock timeout detection */
```

```
i2c_extented_clock_timeout_enable(I2C0);
```

### i2c\_extented\_clock\_timeout\_disable

The description of i2c\_extented\_clock\_timeout\_disable is shown as below:

**Table 3-686. Function i2c\_extented\_clock\_timeout\_disable**

<b>Function name</b>	i2c_extented_clock_timeout_disable
<b>Function prototype</b>	void i2c_extented_clock_timeout_disable(uint32_t i2c_periph);

<b>Function descriptions</b>	disable extended clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable extended clock timeout detection */
i2c_extented_clock_timeout_disable(I2C0);
```

### i2c\_clock\_timeout\_enable

The description of i2c\_clock\_timeout\_enable is shown as below:

**Table 3-687. Function i2c\_clock\_timeout\_enable**

<b>Function name</b>	i2c_clock_timeout_enable
<b>Function prototype</b>	void i2c_clock_timeout_enable(uint32_t i2c_periph);
<b>Function descriptions</b>	enable clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable clock timeout detection */
i2c_clock_timeout_enable(I2C0);
```

### i2c\_clock\_timeout\_disable

The description of i2c\_clock\_timeout\_disable is shown as below:

**Table 3-688. Function i2c\_clock\_timeout\_disable**

<b>Function name</b>	i2c_clock_timeout_disable
<b>Function prototype</b>	void i2c_clock_timeout_disable(uint32_t i2c_periph);

<b>Function descriptions</b>	disable clock timeout detection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable clock timeout detection */
i2c_clock_timeout_disable(I2C0);
```

### i2c\_bus\_timeout\_b\_config

The description of i2c\_bus\_timeout\_b\_config is shown as below:

**Table 3-689. Function i2c\_bus\_timeout\_b\_config**

<b>Function name</b>	i2c_bus_timeout_b_config
<b>Function prototype</b>	void i2c_bus_timeout_b_config(uint32_t i2c_periph, uint32_t timeout);
<b>Function descriptions</b>	configure bus timeout B
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>timeout</b>	0-0xffff, bus timeout B
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure bus timeout B */
i2c_bus_timeout_b_config(I2C0, 0xff);
```

### i2c\_bus\_timeout\_a\_config

The description of i2c\_bus\_timeout\_a\_config is shown as below:

Table 3-690. Function i2c\_bus\_timeout\_a\_config

Function name	i2c_bus_timeout_a_config
Function prototype	void i2c_bus_timeout_a_config(uint32_t i2c_periph, uint32_t timeout);
Function descriptions	configure bus timeout A
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
timeout	0-0xffff, bus timeout A
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure bus timeout A */
i2c_bus_timeout_a_config(I2C0, 0xff);
```

### i2c\_idle\_clock\_timeout\_config

The description of i2c\_idle\_clock\_timeout\_config is shown as below:

Table 3-691. Function i2c\_idle\_clock\_timeout\_config

Function name	i2c_idle_clock_timeout_config
Function prototype	void i2c_idle_clock_timeout_config(uint32_t i2c_periph, uint32_t timeout);
Function descriptions	configure idle clock timeout detection
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
timeout	bus timeout A
BUSTOA_DETECT_SCL_LOW	BUSTOA is used to detect SCL low timeout
BUSTOA_DETECT_IDLE	BUSTOA is used to detect both SCL and SDA high timeout when the bus is idle
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure idle clock timeout detection */
```

```
i2c_idle_clock_timeout_config(I2C0, BUSTOA_DETECT_SCL_LOW);
```

## i2c\_flag\_get

The description of i2c\_flag\_get is shown as below:

**Table 3-692. Function i2c\_flag\_get**

<b>Function name</b>	i2c_flag_get
<b>Function prototype</b>	FlagStatus i2c_flag_get(uint32_t i2c_periph, uint32_t flag);
<b>Function descriptions</b>	get I2C flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>flag</b>	I2C flags
<i>I2C_FLAG_TBE</i>	I2C_TDATA is empty during transmitting
<i>I2C_FLAG_TI</i>	transmit interrupt
<i>I2C_FLAG_RBNE</i>	I2C_RDATA is not empty during receiving
<i>I2C_FLAG_ADDSEND</i>	address received matches in slave mode
<i>I2C_FLAG_NACK</i>	not acknowledge flag
<i>I2C_FLAG_STPDET</i>	STOP condition detected in slave mode
<i>I2C_FLAG_TC</i>	transfer complete in master mode
<i>I2C_FLAG_TCR</i>	transfer complete reload
<i>I2C_FLAG_BERR</i>	bus error
<i>I2C_FLAG_LOSTARB</i>	arbitration Lost
<i>I2C_FLAG_OUERR</i>	overflow/underrun error in slave mode
<i>I2C_FLAG_PECERR</i>	PEC error
<i>I2C_FLAG_TIMEOUT</i>	timeout flag
<i>I2C_FLAG_SMBALT</i>	SMBus Alert
<i>I2C_FLAG_I2CBSY</i>	busy flag
<i>I2C_FLAG_TR</i>	whether the I2C is a transmitter or a receiver in slave mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET / RESET

Example:

```
/* get I2C flag status */
```

```
FlagStatus flag_state = RESET;
```

```
flag_state = i2c_flag_get(I2C0, I2C_FLAG_TBE);
```

## i2c\_flag\_clear

The description of i2c\_flag\_clear is shown as below:

**Table 3-693. Function i2c\_flag\_clear**

<b>Function name</b>	i2c_flag_clear
<b>Function prototype</b>	void i2c_flag_clear(uint32_t i2c_periph, 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>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>flag</b>	I2C flags
<i>I2C_FLAG_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(I2C0, I2C_FLAG_BERR);
```

## i2c\_interrupt\_enable

The description of i2c\_interrupt\_enable is shown as below:

**Table 3-694. Function i2c\_interrupt\_enable**

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

Input parameter{in}	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
<b>interrupt</b>	I2C interrupts
<i>I2C_INT_ERR</i>	error interrupt
<i>I2C_INT_TC</i>	transfer complete interrupt
<i>I2C_INT_STPDET</i>	stop detection interrupt
<i>I2C_INT_NACK</i>	not acknowledge received interrupt
<i>I2C_INT_ADDM</i>	address match interrupt
<i>I2C_INT_RBNE</i>	receive interrupt
<i>I2C_INT_TI</i>	transmit interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable I2C0 transmit interrupt */
```

```
i2c_interrupt_enable(I2C0, I2C_INT_TI);
```

### i2c\_interrupt\_disable

The description of i2c\_interrupt\_disable is shown as below:

**Table 3-695. Function i2c\_interrupt\_disable**

<b>Function name</b>	i2c_interrupt_disable
<b>Function prototype</b>	void i2c_interrupt_disable(uint32_t i2c_periph, uint32_t interrupt);
<b>Function descriptions</b>	disable I2C interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
Input parameter{in}	
<b>interrupt</b>	I2C interrupts
<i>I2C_INT_ERR</i>	error interrupt
<i>I2C_INT_TC</i>	transfer complete interrupt
<i>I2C_INT_STPDET</i>	stop detection interrupt
<i>I2C_INT_NACK</i>	not acknowledge received interrupt
<i>I2C_INT_ADDM</i>	address match interrupt
<i>I2C_INT_RBNE</i>	receive interrupt
<i>I2C_INT_TI</i>	transmit interrupt

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C0 transmit interrupt */
```

```
i2c_interrupt_disable(I2C0, I2C_INT_TI);
```

## i2c\_interrupt\_flag\_get

The description of i2c\_interrupt\_flag\_get is shown as below:

**Table 3-696. Function i2c\_interrupt\_flag\_get**

Function name	i2c_interrupt_flag_get
Function prototype	FlagStatus i2c_interrupt_flag_get(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag);
Function descriptions	get I2C interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
i2c_periph	I2C peripheral
I2Cx	(x=0,1,2,3)
Input parameter{in}	
int_flag	I2C interrupt flags, refer to <a href="#">Table 3-635. 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_PEC RR	PEC error interrupt flag
I2C_INT_FLAG_TIMEO UT	timeout interrupt flag
I2C_INT_FLAG_SMBA	SMBus Alert interrupt flag



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

Example:

```
/* get I2C interrupt flag status */
```

```
FlagStatus flag_state = RESET;
```

```
flag_state = i2c_interrupt_flag_get(I2C0, I2C_INT_FLAG_TI);
```

### i2c\_interrupt\_flag\_clear

The description of i2c\_interrupt\_flag\_clear is shown as below:

**Table 3-697. Function i2c\_interrupt\_flag\_clear**

<b>Function name</b>	i2c_interrupt_flag_clear
<b>Function prototype</b>	void i2c_interrupt_flag_clear(uint32_t i2c_periph, i2c_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	clear I2C interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_periph</b>	I2C peripheral
<i>I2Cx</i>	(x=0,1,2,3)
<b>Input parameter{in}</b>	
<b>int_flag</b>	I2C interrupt flags, refer to <a href="#">Table 3-635. Enum i2c_interrupt_flag_enum.</a>
<i>I2C_INT_FLAG_ADDS</i> <i>END</i>	address received matches in slave mode interrupt flag
<i>I2C_INT_FLAG_NACK</i>	not acknowledge interrupt flag
<i>I2C_INT_FLAG_STPD</i> <i>ET</i>	stop condition detected in slave mode interrupt flag
<i>I2C_INT_FLAG_BERR</i>	bus error interrupt flag
<i>I2C_INT_FLAG_LOSTA</i> <i>RB</i>	arbitration lost interrupt flag
<i>I2C_INT_FLAG_OUER</i> <i>R</i>	overflow/underrun error in slave mode interrupt flag
<i>I2C_INT_FLAG_PEC</i> <i>RR</i>	PEC error interrupt flag
<i>I2C_INT_FLAG_TIMEO</i> <i>UT</i>	timeout interrupt flag
<i>I2C_INT_FLAG_SMBA</i> <i>LT</i>	SMBus Alert interrupt flag

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear a bus error flag */
```

```
i2c_interrupt_flag_clear(I2C0, I2C_INT_FLAG_BERR);
```

## 3.21. LPDTS

Low power digital temperature sensor(LPDTS) is used to transmit square wave,which is converted by temperature and the frequency is proportional to the absolute temperature. The LPDTS registers are listed in chapter [3.21.1](#), the LPDTS firmware functions are introduced in chapter [3.21.2](#).

### 3.21.1. Descriptions of Peripheral registers

**Table 3-698. LPDTS Registers**

Registers	Descriptions
LPDTS_CFG	LPDTS configuration register
LPDTS_SDATA	LPDTS sensor T0 data register
LPDTS_RDATA	LPDTS ramp data register
LPDTS_IT	LPDTS interrupt threshold register
LPDTS_DATA	LPDTS temperature data register
LPDTS_STAT	LPDTS temperature sensor status register
LPDTS_INTEN	LPDTS interrupt enable register
LPDTS_INTC	LPDTS interrupt clear flag register
LPDTS_OP	LPDTS option register

### 3.21.2. Descriptions of Peripheral functions

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

**Table 3-699. LPDTS firmware function**

Function name	Function description
lpdts_deinit	reset LPDTS peripheral
lpdts_struct_para_init	initialize the parameters of LPDTS struct with the default values
lpdts_init	initialize LPDTS peripheral parameter
lpdts_enable	enable LPDTS peripheral
lpdts_disable	disable LPDTS peripheral
lpdts_soft_trigger_enable	enable LPDTS software trigger

Function name	Function description
lpdts_soft_trigger_disable	disable LPDTS software trigger
lpdts_high_threshold_set	configure LPDTS interrupt high threshold
lpdts_low_threshold_set	configure LPDTS interrupt low threshold
lpdts_ref_clock_source_config	configure LPDTS clock source
lpdts_temperature_get	get LPDTS collection temperature value
lpdts_flag_get	get LPDTS status flag
lpdts_interrupt_enable	enable LPDTS interrupt
lpdts_interrupt_disable	disable LPDTS interrupt
lpdts_interrupt_flag_get	get LPDTS interrupt flag status
lpdts_interrupt_flag_clear	clear LPDTS interrupt flag status

### Structure lpdts\_parameter\_struct

**Table3-700. Structure lpdts\_parameter\_struct**

Member name	Function description
ref_clock	Reference clock selection (REF_PCLK, REF_LXTAL)
trigger_input	Input trigger source selection (NO_HARDWARE_TRIGGER, LPDTS_TRG)
sampling_time	Sampling time setting (SPT_CLOCK_1, SPT_CLOCK_2, SPT_CLOCK_3, SPT_CLOCK_4, SPT_CLOCK_5, SPT_CLOCK_6, SPT_CLOCK_7, SPT_CLOCK_8, SPT_CLOCK_9, SPT_CLOCK_10, SPT_CLOCK_2, SPT_CLOCK_11, SPT_CLOCK_12, SPT_CLOCK_13, SPT_CLOCK_14, SPT_CLOCK_15)

### lpdts\_deinit

The description of lpdts\_deinit is shown as below:

**Table3-701. Function lpdts\_deinit**

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

Example:

```
/* reset the LPDTS registers */
```

```
lpdts_deinit();
```

## lpdts\_struct\_para\_init

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

**Table3-702. Function lpdts\_struct\_para\_init**

<b>Function name</b>	lpdts_struct_para_init
<b>Function prototype</b>	void lpdts_struct_para_init(lpdts_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize the parameters of LPDTS struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>init_struct</b>	pointer to a lpdts_parameter_struct structure which contains parameters for initialization of the lpdts peripheral, the structure members can refer to members of the structure <a href="#">Table3-700. Structure lpdts_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of LPDTS */
lpdts_parameter_struct lpdts_init_struct;
lpdts_struct_para_init(&lpdts_init_struct);
```

## lpdts\_init

The description of lpdts\_init is shown as below:

**Table3-703. Function lpdts\_init**

<b>Function name</b>	lpdts_init
<b>Function prototype</b>	void lpdts_init(lpdts_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize LPDTS peripheral parameter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>init_struct</b>	pointer to a lpdts_parameter_struct structure which contains parameters for initialization of the lpdts peripheral, the structure members can refer to members of the structure <a href="#">Table3-700. Structure lpdts_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the LPDTS */

lpdts_parameter_struct lpdts_init_struct;

lpdts_init(&lpdts_init_struct);
```

### lpdts\_enable

The description of lpdts\_enable is shown as below:

**Table3-704. Function lpdts\_enable**

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

Example:

```
/* enable LPDTS temperature sensor */

lpdts_enable();
```

### lpdts\_disable

The description of lpdts\_disable is shown as below:

**Table3-705. Function lpdts\_disable**

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

Example:

```
/* disable LPDTS temperature sensor */
```

```
lpdts_disable();
```

### lpdts\_soft\_trigger\_enable

The description of lpdts\_soft\_trigger\_enable is shown as below:

**Table3-706. Function lpdts\_soft\_trigger\_enable**

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

Example:

```
/* software trigger start */
```

```
lpdts_soft_trigger_enable();
```

### lpdts\_soft\_trigger\_disable

The description of lpdts\_soft\_trigger\_disable is shown as below:

**Table3-707. Function lpdts\_soft\_trigger\_disable**

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

Example:

```
/* disable LPDTS software trigger */
```

```
lpdts_soft_trigger_disable();
```

## lpdts\_high\_threshold\_set

The description of lpdts\_high\_threshold\_set is shown as below:

**Table3-708. Function lpdts\_high\_threshold\_set**

<b>Function name</b>	lpdts_high_threshold_set
<b>Function prototype</b>	void lpdts_high_threshold_set(uint16_t value);
<b>Function descriptions</b>	configure LPDTS interrupt high threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>value</b>	interrupt high threshold
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set threshold value */
lpdts_high_threshold_set(0U);
```

## lpdts\_low\_threshold\_set

The description of lpdts\_low\_threshold\_set is shown as below:

**Table3-709. Function lpdts\_low\_threshold\_set**

<b>Function name</b>	lpdts_low_threshold_set
<b>Function prototype</b>	void lpdts_low_threshold_set(uint16_t value);
<b>Function descriptions</b>	configure LPDTS interrupt low threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>value</b>	interrupt low threshold
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set threshold value */
lpdts_low_threshold_set(0U);
```

## lpdts\_ref\_clock\_source\_config

The description of lpdts\_ref\_clock\_source\_config is shown as below:

Table3-710. Function `lpdts_ref_clock_source_config`

Function name	<code>lpdts_ref_clock_source_config</code>
Function prototype	<code>void lpdts_ref_clock_source_config(uint32_t source);</code>
Function descriptions	configure LPDTS clock source
Precondition	-
The called functions	-
Input parameter{in}	
<b>source</b>	clock source
<i>REF_PCLK</i>	high speed reference clock
<i>REF_LXTAL</i>	low speed reference clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select reference clock */
```

```
lpdts_ref_clock_source_config(REF_PCLK);
```

### `lpdts_temperature_get`

The description of `lpdts_temperature_get` is shown as below:

Table3-711. Function `lpdts_temperature_get`

Function name	<code>lpdts_temperature_get</code>
Function prototype	<code>int32_t lpdts_temperature_get(void);</code>
Function descriptions	get LPDTS collection temperature value
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<b>int32_t</b>	The resulting temperature value

Example:

```
int32_t temperature;
```

```
/* wait till TSRF flag is set */
```

```
while(RESET == lpdts_flag_get(LPDTS_FLAG_TSRF)){
```

```
}
```

```
temperature = lpdts_temperature_get();
```



## lpdts\_flag\_get

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

**Table3-712. Function lpdts\_flag\_get**

<b>Function name</b>	lpdts_flag_get
<b>Function prototype</b>	FlagStatus lpdts_flag_get(uint32_t flag);
<b>Function descriptions</b>	get LPDTS status flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>flag</b>	interrupt flag
<i>LPDTS_FLAG_TSR</i>	temperature sensor ready flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* wait till TSRF flag is set */
while(RESET == lpdts_flag_get(LPDTTS_FLAG_TSR)) {
}
```

## lpdts\_interrupt\_enable

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

**Table3-713. Function lpdts\_interrupt\_enable**

<b>Function name</b>	lpdts_interrupt_enable
<b>Function prototype</b>	void lpdts_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable LPDTS interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	interrupt type
<i>LPDTS_INT_EM</i>	end of measurement interrupt (synchronized on PCLK)
<i>LPDTS_INT_LT</i>	low threshold interrupt (synchronized on PCLK)
<i>LPDTS_INT_HT</i>	high threshold interrupt (synchronized on PCLK)
<i>LPDTS_INT_EMA</i>	end of measurement asynchronous interrupt
<i>LPDTS_INT_LTA</i>	low threshold asynchronous interrupt
<i>LPDTS_INT-HTA</i>	high threshold asynchronous interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* enable end of measurement interrupt */
```

```
lpdts_interrupt_enable(LPPTS_INT_EM);
```

### lpdts\_interrupt\_disable

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

**Table3-714. Function lpdts\_interrupt\_disable**

<b>Function name</b>	lpdts_interrupt_disable
<b>Function prototype</b>	void lpdts_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disable LPDTS interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	interrupt type
<i>LPDTS_INT_EM</i>	end of measurement interrupt (synchronized on PCLK)
<i>LPDTS_INT_LT</i>	low threshold interrupt (synchronized on PCLK)
<i>LPDTS_INT_HT</i>	high threshold interrupt (synchronized on PCLK)
<i>LPDTS_INT_EMA</i>	end of measurement asynchronous interrupt
<i>LPDTS_INT_LTA</i>	low threshold asynchronous interrupt
<i>LPDTS_INT-HTA</i>	high threshold asynchronous interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable end of measurement interrupt */
```

```
lpdts_interrupt_disable(LPPTS_INT_EM);
```

### lpdts\_interrupt\_flag\_get

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

**Table3-715. Function lpdts\_interrupt\_flag\_get**

<b>Function name</b>	lpdts_interrupt_flag_get
<b>Function prototype</b>	FlagStatus lpdts_interrupt_flag_get(uint32_t flag);
<b>Function descriptions</b>	get LPDTS interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

flag	interrupt flag
LPDTS_INT_FLAG_EM	end of measurement interrupt flag
LPDTS_INT_FLAG_LT	low threshold interrupt flag
LPDTS_INT_FLAG_HT	high threshold interrupt flag
LPDTS_INT_FLAG_EMA	end of measurement asynchronous interrupt flag
LPDTS_INT_FLAG_LTA	low threshold asynchronous interrupt flag
LPDTS_INT_FLAG_HTA	high threshold asynchronous interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* wait till HTA flag is set */
while(RESET == lpdts_interrupt_flag_get(LPDTS_INT_FLAG_HTA)) {
}
```

### lpdts\_interrupt\_flag\_clear

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

**Table3-716. Function lpdts\_interrupt\_flag\_clear**

Function name	lpdts_interrupt_flag_clear
Function prototype	void lpdts_interrupt_flag_clear(uint32_t flag);
Function descriptions	clear LPDTS interrupt flag status
Precondition	-
The called functions	-
<b>Input parameter{in}</b>	
flag	interrupt flag
LPDTS_INT_FLAG_EM	end of measurement interrupt flag
LPDTS_INT_FLAG_LT	low threshold interrupt flag
LPDTS_INT_FLAG_HT	high threshold interrupt flag
LPDTS_INT_FLAG_EMA	end of measurement asynchronous interrupt flag
LPDTS_INT_FLAG_LTA	low threshold asynchronous interrupt flag
LPDTS_INT_FLAG_HTA	high threshold asynchronous interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear HTA flag */
lpdts_interrupt_flag_clear(LPDTS_INT_FLAG_HTA);
```

## 3.22. MDMA

The master direct memory access controller (MDMA) 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 MDMA registers are listed in chapter [3.22.1](#), the MDMA firmware functions are introduced in chapter [3.22.2](#).

### 3.22.1. Descriptions of Peripheral registers

DMA registers are listed in the table shown as below:

**Table 3-717. MDMA Registers**

Registers	Descriptions
MDMA_GINTF	global interrupt flag register
MDMA_CHxSTAT0 (x=0..15)	channel x status register 0
MDMA_CHxSTATC (x=0..15)	channel x status clear register
MDMA_CHxSTAT1 (x=0..15)	channel x status register 1
MDMA_CHxCTL0 (x=0..15)	channel x control register 0
MDMA_CHxCFG (x=0..15)	channel x configure register
MDMA_CHxBTCFG (x=0..15)	channel x block transfer configure register
MDMA_CHxSADDR (x=0..15)	channel x source address register
MDMA_CHxDADDR (x=0..15)	channel x destination address register
MDMA_CHxMBAD DRU (x=0..15)	channel x multi-block address update register
MDMA_CHxLADDR (x=0..15)	channel x link address register
MDMA_CHxCTL1 (x=0..15)	channel x control register 1
MDMA_CHxMADDR R (x=0..15)	channel x mask address register
MDMA_CHxMDATA (x=0..15)	channel x mask data register

### 3.22.2. Descriptions of Peripheral functions

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

**Table 3-718. MDMA firmware function**

Function name	Function description
mdma_deinit	deinitialize MDMA
mdma_channel_deinit	deinitialize MDMA registers of a channel
mdma_para_struct_init	initialize the MDMA parameters struct with the default values
mdma_multi_block_para_struct_init	initialize the MDMA multi block transfer mode parameters struct with the default values
mdma_link_node_para_struct_init	initialize the MDMA link node configuration struct with the default values
mdma_init	initialize MDMA channel with MDMA parameter structure
mdma_buffer_block_mode_config	configure MDMA buffer/block transfer mode
mdma_multi_block_mode_config	configure MDMA multi block transfer mode
mdma_node_create	create MDMA link list node
mdma_node_add	MDMA add node to link list
mdma_node_delete	MDMA disconnect link list node
mdma_destination_address_config	configure MDMA destination base address
mdma_source_address_config	configure MDMA source base address
mdma_destination_bus_config	configure MDMA destination bus
mdma_source_bus_config	configure MDMA source bus
mdma_priority_config	configure priority level of MDMA channel
mdma_endianness_config	configure endianness of MDMA channel
mdma_alignment_config	configure data alignment of MDMA channel
mdma_source_burst_beats_config	configure transfer burst beats of source
mdma_destination_burst_beats_config	configure transfer burst beats of destination
mdma_source_width_config	configure data size of source
mdma_destination_width_config	configure data size of destination
mdma_source_increment_config	configure source address increment mode
mdma_destination_increment_config	configure destination address increment mode
mdma_channel_bufferable_write_enable	enable MDMA channel bufferable write mode
mdma_channel_bufferable_write_disable	disable MDMA channel bufferable write mode
mdma_channel_software_request_enable	enable MDMA channel software request
mdma_channel_enable	enable MDMA channel
mdma_channel_disable	disable MDMA channel
mdma_transfer_error_direction_get	get MDMA transfer error direction
mdma_transfer_error_address_get	get MDMA transfer error address

Function name	Function description
mdma_flag_get	get MDMA flag
mdma_flag_clear	clear MDMA flag
mdma_interrupt_enable	enable MDMA interrupt
mdma_interrupt_disable	disable MDMA interrupt
mdma_interrupt_flag_get	get MDMA interrupt flag
mdma_interrupt_flag_clear	clear MDMA interrupt flag

### Structure mdma\_parameter\_struct

**Table 3-719. Structure mdma\_parameter\_struct**

Member name	Function description
request	specifies the MDMA request
trans_trig_mode	specifies the trigger transfer mode
priority	specifies the software priority for the MDMA channelx
endianness	specifies if the MDMA transactions preserve the little endianness
source_inc	specifies the source increment mode
dest_inc	specifies the destination increment mode
source_data_size	specifies the source data size
dest_data_dize	specifies the destination data size
data_alignment	specifies the source to destination memory data packing/padding mode
buff_trans_len	specifies the buffer transfer length (number of bytes)
source_burst	specifies the burst transfer configuration for the source memory transfers
dest_burst	specifies the burst transfer configuration for the destination memory transfers
mask_addr	mask address
mask_data	mask data
source_addr	specifies the source address
dest_addr	specifies the destination address
tbytes_num_in_block	specifies the transfer bytes number in a buffer or block transfer
source_bus	specifies the source bus
dest_bus	specifies the destination bus
bufferable_write_mode	specifies the bufferable write mode

### Structure mdma\_multi\_block\_parameter\_struct

**Table 3-720. Structure mdma\_multi\_block\_parameter\_struct**

Member name	Function description
block_num	multi-block number
saddr_update_val	source address update value
dstaddr_update_val	destination address update value
saddr_update_dir	source address update direction
dstaddr_update_dir	destination address update direction

## Structure mdma\_link\_node\_parameter\_struct

**Table 3-721. Structure mdma\_link\_node\_parameter\_struct**

Member name	Function description
chxcfg_reg	channel x configure register
chxbtcfg_reg	channel x block transfer configure register
chxsaddr_reg	channel x source address register
chxdaddr_reg	channel x destination address register
chxmbaddru_reg	channel x multi-block address update register
chxladdr_reg	channel x link address register
chxctl1_reg	channel x control register 1
reserved	channel x reserved register
chxmaddr_reg	channel x mask address register
chxmdata_reg	channel x mask data register

## Enum mdma\_add\_update\_dir\_enum

**Table 3-722. Enum mdma\_add\_update\_dir\_enum**

Member name	Function description
UPDATE_DIR_INC REASE	MDMA address update increase
UPDATE_DIR_DEC REASE	MDMA address update decrease

## Enum mdma\_channel\_enum

**Table 3-723. Enum mdma\_channel\_enum**

Member name	Function description
MDMA_CH0	MDMA channel 0
MDMA_CH1	MDMA channel 1
MDMA_CH2	MDMA channel 2
MDMA_CH3	MDMA channel 3
MDMA_CH4	MDMA channel 4
MDMA_CH5	MDMA channel 5
MDMA_CH6	MDMA channel 6
MDMA_CH7	MDMA channel 7
MDMA_CH8	MDMA channel 8
MDMA_CH9	MDMA channel 9
MDMA_CH10	MDMA channel 10
MDMA_CH11	MDMA channel 11
MDMA_CH12	MDMA channel 12
MDMA_CH13	MDMA channel 13
MDMA_CH14	MDMA channel 14
MDMA_CH15	MDMA channel 15

## mdma\_deinit

The description of mdma\_deinit is shown as below:

**Table 3-724. Function mdma\_deinit**

<b>Function name</b>	mdma_deinit
<b>Function prototype</b>	void mdma_deinit(void);
<b>Function descriptions</b>	deinitialize MDMA
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize MDMA */
mdma_deinit();
```

## mdma\_channel\_deinit

The description of mdma\_channel\_deinit is shown as below:

**Table 3-725. Function mdma\_channel\_deinit**

<b>Function name</b>	mdma_channel_deinit
<b>Function prototype</b>	void mdma_channel_deinit(mdma_channel_enum channelx);
<b>Function descriptions</b>	deinitialize MDMA registers of a channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize MDMA channel 0 */
mdma_channel_deinit(MDMA_CH0);
```



## mdma\_para\_struct\_init

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

**Table 3-726. Function mdma\_para\_struct\_init**

<b>Function name</b>	mdma_para_struct_init
<b>Function prototype</b>	void mdma_para_struct_init(mdma_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize the MDMA parameters struct 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 MDMA channel, refer to <a href="#">Table 3-719. Structure mdma_parameter_struct.</a>
<b>Return value</b>	
-	-

Example:

```
/* initialize the MDMA single data mode parameters struct with the default values */
mdma_parameter_struct mdma_init_struct;
mdma_para_struct_init(&mdma_init_struct);
```

## mdma\_multi\_block\_para\_struct\_init

The description of mdma\_multi\_block\_para\_struct\_init is shown as below:

**Table 3-727. Function mdma\_multi\_block\_para\_struct\_init**

<b>Function name</b>	mdma_multi_block_para_struct_init
<b>Function prototype</b>	void mdma_multi_block_para_struct_init (mdma_multi_block_parameter_struct *block_init_struct);
<b>Function descriptions</b>	initialize the MDMA multi block transfer mode parameters struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>block_init_struct</b>	the initialization data needed to initialize MDMA channel, refer to <a href="#">Table 3-720. Structure mdma_multi_block_parameter_struct.</a>
<b>Return value</b>	
-	-

Example:

```

/* initialize the MDMA multi block transfer mode parameters struct with the default values */

mdma_multi_block_parameter_struct mdma_multi_block_init_struct;

mdma_multi_block_para_struct_init (&mdma_multi_block_init_struct);

```

### mdma\_link\_node\_para\_struct\_init

The description of mdma\_link\_node\_para\_struct\_init is shown as below:

**Table 3-728. Function mdma\_link\_node\_para\_struct\_init**

<b>Function name</b>	mdma_link_node_para_struct_init
<b>Function prototype</b>	void mdma_link_node_para_struct_init(mdma_link_node_parameter_struct *node);
<b>Function descriptions</b>	initialize the MDMA link node configuration struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>node</b>	initialize the MDMA link node configuration struct with the default values, refer to <a href="#">Table 3-721. Structure mdma_link_node_parameter_struct</a> .
<b>Return value</b>	
-	-

Example:

```

/* initialize the MDMA link node configuration struct with the default values */

mdma_link_node_parameter_struct node;

mdma_link_node_para_struct_init (&node);

```

### mdma\_init

The description of mdma\_init is shown as below:

**Table 3-729. Function mdma\_init**

<b>Function name</b>	mdma_init
<b>Function prototype</b>	void mdma_init(mdma_channel_enum channelx, mdma_parameter_struct *init_struct);
<b>Function descriptions</b>	initialize MDMA channel with MDMA parameter structure
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .

Input parameter{in}	
<b>init_struct</b>	Structure for initialization, the structure members can refer to <a href="#">Table 3-719. Structure mdma_parameter_struct.</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* initialize MDMA channel 0 */

mdma_para_struct_init(&mdma_init_struct);

mdma_init_struct.request = MDMA_REQUEST_DMA1_CHANNEL2_FTFIF;

mdma_init_struct.transfer_trigger_mode = MDMA_BUFFER_TRANSFER;

mdma_init_struct.priority = MDMA_PRIORITY_HIGH;

mdma_init_struct.endianness = MDMA_LITTLE_ENDIANNESS;

mdma_init_struct.source_addr = (uint32_t)&usart_rxbuffer;

mdma_init_struct.dest_addr = mdma_rxbuffer_addr;

mdma_init_struct.source_inc = MDMA_SOURCE_INCREASE_8BIT;

mdma_init_struct.dest_inc = MDMA_DESTINATION_INCREASE_8BIT;

mdma_init_struct.source_data_size = MDMA_SOURCE_DATASIZE_8BIT;

mdma_init_struct.dest_data_dize = MDMA_DESTINATION_DATASIZE_8BIT;

mdma_init_struct.source_burst = MDMA_SOURCE_BURST_SINGLE;

mdma_init_struct.dest_burst = MDMA_DESTINATION_BURST_SINGLE;

mdma_init_struct.source_bus = MDMA_SOURCE_AXI;

mdma_init_struct.dest_bus = MDMA_DESTINATION_AHB_TCM;

mdma_init_struct.data_alignment = MDMA_DATAALIGN_PKEN;

mdma_init_struct.buffertransfer_length = 10U;

mdma_init_struct.tbytes_num_in_block = 10U;

mdma_init_struct.mask_addr = DMA1_INTC_ADDRESS;

mdma_init_struct.mask_data = DMA1_INTC_FTFIFC2;

mdma_init(MDMA_CH0, &mdma_init_struct);

```

## mdma\_buffer\_block\_mode\_config

The description of mdma\_buffer\_block\_mode\_config is shown as below:

**Table 3-730. Function mdma\_buffer\_block\_mode\_config**

<b>Function name</b>	mdma_buffer_block_mode_config
<b>Function prototype</b>	void mdma_buffer_block_mode_config(mdma_channel_enum channelx, uint32_t saddr, uint32_t daddr, uint32_t tbnun);
<b>Function descriptions</b>	configure MDMA buffer/block transfer mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>saddr</b>	source address, 0x00000000-0xFFFFFFFF
<b>Input parameter{in}</b>	
<b>daddr</b>	destination address, 0x00000000-0xFFFFFFFF
<b>Input parameter{in}</b>	
<b>tbnun</b>	number of bytes to transfer, 0x00000000-0x00010000
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure MDMA buffer/block transfer mode */
mdma_buffer_block_mode_config(0x08000000, 0x24100000, 256);
```

## mdma\_multi\_block\_mode\_config

The description of mdma\_multi\_block\_mode\_config is shown as below:

**Table 3-731. Function mdma\_multi\_block\_mode\_config**

<b>Function name</b>	mdma_multi_block_mode_config
<b>Function prototype</b>	void mdma_multi_block_mode_config(mdma_channel_enum channelx, uint32_t tbnun, mdma_multi_block_parameter_struct *block_init_struct);
<b>Function descriptions</b>	configure MDMA multi block transfer mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum</a>

	<a href="#"><u>mdma_channel_enum.</u></a>
<b>Input parameter{in}</b>	
<b>tbnum</b>	number of bytes to transfer in block, 0x00000000-0x00000FFF
<b>Input parameter{in}</b>	
<b>block_init_struct</b>	Structure for MDMA multi block mode, the structure members can refer to <a href="#"><u>Table 3-720. Structure mdma_multi_block_parameter_struct.</u></a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure MDMA multi block transfer mode */
mdma_multi_block_parameter_struct block_init_struct;
mdma_multi_block_mode_config(MDMA_CH0, 0xFF, & block_init_struct);
```

### mdma\_node\_create

The description of mdma\_node\_create is shown as below:

**Table 3-732. Function mdma\_node\_create**

<b>Function name</b>	mdma_node_create
<b>Function prototype</b>	void mdma_node_create(mdma_link_node_parameter_struct *node, mdma_multi_block_parameter_struct *block_init_struct, mdma_parameter_struct *init_struct);
<b>Function descriptions</b>	MDMA link list node create
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>block_init_struct</b>	Structure for MDMA multi block mode, the structure members can refer to <a href="#"><u>Table 3-720. Structure mdma_multi_block_parameter_struct.</u></a>
<b>Input parameter{in}</b>	
<b>init_struct</b>	Structure for initialization, the structure members can refer to <a href="#"><u>Table 3-719. Structure mdma_parameter_struct.</u></a>
<b>Output parameter{out}</b>	
<b>node</b>	Structure for link node, the structure members can refer to <a href="#"><u>Table 3-721. Structure mdma_link_node_parameter_struct.</u></a>
<b>Return value</b>	
-	-

Example:

```
/* create MDMA link list node */
mdma_link_node_parameter_struct node;
```

```
mdma_multi_block_parameter_struct block_init_struct;

mdma_parameter_struct init_struct;

mdma_node_create(&node, &block_init_struct, &init_struct);
```

## mdma\_node\_add

The description of mdma\_node\_add is shown as below:

**Table 3-733. Function mdma\_node\_add**

Function name	mdma_node_add
Function prototype	void mdma_node_add(mdma_link_node_parameter_struct *pre_node, mdma_link_node_parameter_struct *new_node);
Function descriptions	MDMA add node to link list
Precondition	-
The called functions	-
Input parameter{in}	
pre_node	Structure for previous node, the structure members can refer to <a href="#">Table 3-721. Structure mdma link node parameter struct.</a>
Input parameter{in}	
new_node	Structure for new node, the structure members can refer to <a href="#">Table 3-721. Structure mdma link node parameter struct.</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* MDMA add node to link list */

mdma_link_node_parameter_struct pre_node;

mdma_link_node_parameter_struct new_node;

mdma_node_add(&pre_node, &new_node);
```

## mdma\_node\_delete

The description of mdma\_node\_delete is shown as below:

**Table 3-734. Function mdma\_node\_delete**

Function name	mdma_node_delete
Function prototype	ErrStatus mdma_node_delete(mdma_link_node_parameter_struct *pre_node, mdma_link_node_parameter_struct *unused_node);
Function descriptions	MDMA disconnect link list node
Precondition	-
The called functions	-

Input parameter{in}	
<b>pre_node</b>	Structure for previous node, the structure members can refer to <a href="#">Table 3-721. Structure mdma link node parameter struct.</a>
Input parameter{in}	
<b>new_node</b>	Structure for new node, the structure members can refer to <a href="#">Table 3-721. Structure mdma link node parameter struct.</a>
Output parameter{out}	
-	-
Return value	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

```
/* MDMA disconnect link list node */
mdma_link_node_parameter_struct pre_node;
mdma_link_node_parameter_struct unused_node;
mdma_node_delete(&pre_node, &unused_node);
```

### mdma\_destination\_address\_config

The description of mdma\_destination\_address\_config is shown as below:

**Table 3-735. Function mdma\_destination\_address\_config**

<b>Function name</b>	mdma_destination_address_config
<b>Function prototype</b>	void mdma_destination_address_config(mdma_channel_enum channelx, uint32_t address);
<b>Function descriptions</b>	configure MDMA destination base address
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum.</a>
Input parameter{in}	
<b>address</b>	destination base address, 0x00000000-0xFFFFFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure MDMA destination base address */
mdma_destination_address_config(MDMA_CH0, 0x08000000);
```

## mdma\_source\_address\_config

The description of mdma\_source\_address\_config is shown as below:

**Table 3-736. Function mdma\_source\_address\_config**

<b>Function name</b>	mdma_source_address_config
<b>Function prototype</b>	void mdma_source_address_config(mdma_channel_enum channelx, uint32_t address);
<b>Function descriptions</b>	configure MDMA source base address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>address</b>	Source base address, 0x00000000-0xFFFFFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure MDMA source base address */
```

```
mdma_source_address_config(MDMA_CH0, 0x20000000);
```

## mdma\_destination\_bus\_config

The description of mdma\_destination\_bus\_config is shown as below:

**Table 3-737. Function mdma\_destination\_bus\_config**

<b>Function name</b>	mdma_destination_bus_config
<b>Function prototype</b>	void mdma_destination_bus_config(mdma_channel_enum channelx, uint32_t bus);
<b>Function descriptions</b>	configure MDMA destination bus
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>bus</b>	destination bus
MDMA_DESTINATION	destination bus is the system bus or AXI bus



<i>_AXI</i>	
<i>MDMA_DESTINATION_AHB_TCM</i>	destination bus is AHB bus or TCM
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure MDMA destination bus */
```

```
mdma_destination_bus_config(MDMA_CH0, MDMA_DESTINATION_AHB_TCM);
```

### mdma\_source\_bus\_config

The description of mdma\_source\_bus\_config is shown as below:

**Table 3-738. Function mdma\_source\_bus\_config**

<b>Function name</b>	mdma_source_bus_config
<b>Function prototype</b>	void mdma_source_bus_config(mdma_channel_enum channelx, uint32_t bus);
<b>Function descriptions</b>	configure MDMA source bus
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>bus</b>	source bus
<i>MDMA_SOURCE_AXI</i>	source bus is the system bus or AXI bus
<i>MDMA_SOURCE_AHB_TCM</i>	source bus is AHB bus or TCM
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure MDMA source bus */
```

```
mdma_source_bus_config(MDMA_CH0, MDMA_SOURCE_AHB_TCM);
```

### mdma\_priority\_config

The description of mdma\_priority\_config is shown as below:

**Table 3-739. Function mdma\_priority\_config**

<b>Function name</b>	mdma_priority_config
<b>Function prototype</b>	void mdma_priority_config(mdma_channel_enum channelx, uint32_t priority);
<b>Function descriptions</b>	configure priority level of MDMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>priority</b>	priority Level of this channel
MDMA_PRIORITY_LOW	low priority
MDMA_PRIORITY_MEDIUM	medium priority
MDMA_PRIORITY_HIGH	high priority
MDMA_PRIORITY_ULTRA_HIGH	ultra high priority
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure priority level of MDMA channel */
```

```
mdma_priority_config(MDMA_CH0, MDMA_PRIORITY_ULTRA_HIGH);
```

### mdma\_endianness\_config

The description of mdma\_endianness\_config is shown as below:

**Table 3-740. Function mdma\_endianness\_config**

<b>Function name</b>	mdma_endianness_config
<b>Function prototype</b>	void mdma_endianness_config(mdma_channel_enum channelx, uint32_t endianness);
<b>Function descriptions</b>	configure endianness of MDMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum</a>

	<a href="#"><u>mdma_channel_enum.</u></a>
<b>Input parameter{in}</b>	
<b>endianness</b>	MDMA endianness
<i>MDMA_LITTLE_ENDIANNESS</i>	little endianness
<i>MDMA_BYTE_ENDIANNESS_EXCHANGE</i>	exchange the order of the bytes in a half-word
<i>MDMA_HALFWORD_ENDIANNESS_EXCHANGE</i>	exchange the order of the half-words in a word
<i>MDMA_WORD_ENDIANNESS_EXCHANGE</i>	exchange the order of the words in a double word
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure endianness of MDMA channel */
```

```
mdma_endianness_config(MDMA_CH0, MDMA_BYTE_ENDIANNESS_EXCHANGE);
```

### mdma\_alignment\_config

The description of mdma\_alignment\_config is shown as below:

**Table 3-741. Function mdma\_alignment\_config**

<b>Function name</b>	mdma_alignment_config
<b>Function prototype</b>	void mdma_endianness_config(mdma_channel_enum channelx, uint32_t endianness);
<b>Function descriptions</b>	configure data alignment of MDMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#"><u>Table 3-723. Enum mdma_channel_enum.</u></a>
<b>Input parameter{in}</b>	
<b>alignment</b>	MDMA data alignment
<i>MDMA_DATAALIGN_PACK</i>	pack/unpack the source data to match the destination data size
<i>MDMA_DATAALIGN_RIGHT</i>	right aligned, padded with 0s (default)
<i>MDMA_DATAALIGN_RIGHT_SIGNED</i>	right aligned with sign extended, note: this mode is allowed only if the source data size is smaller than destination data size

<i>MDMA_DATAALIGN_LEFT</i>	left aligned, padded with 0s in low bytes position when source data size smaller than destination data size, and only high byte of source is written when source data size larger than destination data size
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data alignment of MDMA channel */
```

```
mdma_alignment_config(MDMA_CH0, MDMA_DATAALIGN_RIGHT);
```

### mdma\_source\_burst\_beats\_config

The description of mdma\_source\_burst\_beats\_config is shown as below:

**Table 3-742. Function mdma\_source\_burst\_beats\_config**

<b>Function name</b>	mdma_source_burst_beats_config
<b>Function prototype</b>	void mdma_source_burst_beats_config(mdma_channel_enum channelx, uint32_t sbeat);
<b>Function descriptions</b>	configure transfer burst beats of source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>sbeat</b>	source transfer burst beats
<i>MDMA_SOURCE_BURST_SINGLE</i>	single burst
<i>MDMA_SOURCE_BURST_2BEATS</i>	2-beat incrementing burst
<i>MDMA_SOURCE_BURST_4BEATS</i>	4-beat incrementing burst
<i>MDMA_SOURCE_BURST_8BEATS</i>	8-beat incrementing burst
<i>MDMA_SOURCE_BURST_16BEATS</i>	16-beat incrementing burst
<i>MDMA_SOURCE_BURST_32BEATS</i>	32-beat incrementing burst
<i>MDMA_SOURCE_BURST_64BEATS</i>	64-beat incrementing burst
<i>MDMA_SOURCE_BURST_128BEATS</i>	128-beat incrementing burst

ST_128BEATS	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure transfer burst beats of source */
```

```
mdma_source_burst_beats_config(MDMA_CH0, MDMA_SOURCE_BURST_4BEATS);
```

### mdma\_destination\_burst\_beats\_config

The description of mdma\_destination\_burst\_beats\_config is shown as below:

**Table 3-743. Function mdma\_destination\_burst\_beats\_config**

<b>Function name</b>	mdma_destination_burst_beats_config
<b>Function prototype</b>	void mdma_destination_burst_beats_config(mdma_channel_enum channelx, uint32_t dbeat);
<b>Function descriptions</b>	configure transfer burst beats of destination
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>dbeat</b>	destination transfer burst beats
MDMA_DESTINATION_BURST_SINGLE	single burst
MDMA_DESTINATION_BURST_2BEATS	2-beat incrementing burst
MDMA_DESTINATION_BURST_4BEATS	4-beat incrementing burst
MDMA_DESTINATION_BURST_8BEATS	8-beat incrementing burst
MDMA_DESTINATION_BURST_16BEATS	16-beat incrementing burst
MDMA_DESTINATION_BURST_32BEATS	32-beat incrementing burst
MDMA_DESTINATION_BURST_64BEATS	64-beat incrementing burst
MDMA_DESTINATION_BURST_128BEATS	128-beat incrementing burst
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure transfer burst beats of destination */
```

```
mdma_destination_burst_beats_config(MDMA_CH0,
MDMA_DESTINATION_BURST_4BEATS);
```

### mdma\_source\_width\_config

The description of mdma\_source\_width\_config is shown as below:

**Table 3-744. Function mdma\_source\_width\_config**

<b>Function name</b>	mdma_source_width_config
<b>Function prototype</b>	void mdma_source_width_config(mdma_channel_enum channelx, uint32_t swidth);
<b>Function descriptions</b>	configure data size of source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>swidth</b>	source data size
MDMA_SOURCE_DAT ASIZE_8BIT	data size of source is 8-bit
MDMA_SOURCE_DAT ASIZE_16BIT	data size of source is 16-bit
MDMA_SOURCE_DAT ASIZE_32BIT	data size of source is 32-bit
MDMA_SOURCE_DAT ASIZE_64BIT	data size of source is 64-bit
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data size of source */
```

```
mdma_source_width_config (MDMA_CH0, MDMA_SOURCE_DATASIZE_8BIT);
```

## mdma\_destination\_width\_config

The description of mdma\_destination\_width\_config is shown as below:

**Table 3-745. Function mdma\_destination\_width\_config**

<b>Function name</b>	mdma_destination_width_config
<b>Function prototype</b>	void mdma_destination_width_config(mdma_channel_enum channelx, uint32_t dwidth);
<b>Function descriptions</b>	configure data size of destination
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>dwidth</b>	destination data size
MDMA_DESTINATION_DATASIZE_8BIT	data size of destination is 8-bit
MDMA_DESTINATION_DATASIZE_16BIT	data size of destination is 16-bit
MDMA_DESTINATION_DATASIZE_32BIT	data size of destination is 32-bit
MDMA_DESTINATION_DATASIZE_64BIT	data size of destination is 64-bit
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data size of destination */
```

```
mdma_destination_width_config (MDMA_CH0, MDMA_DESTINATION_DATASIZE_16BIT);
```

## mdma\_source\_increment\_config

The description of mdma\_source\_increment\_config is shown as below:

**Table 3-746. Function mdma\_source\_increment\_config**

<b>Function name</b>	mdma_source_increment_config
<b>Function prototype</b>	void mdma_source_increment_config(mdma_channel_enum channelx, uint32_t sinc);
<b>Function descriptions</b>	configure source adress increment mode
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum <i>mdma_channel_enum</i></a> .
Input parameter{in}	
<b>sinc</b>	source address increment mode
<i>MDMA_SOURCE_INC REASE_DISABLE</i>	no increment
<i>MDMA_SOURCE_INC REASE_8BIT</i>	source address is increased by 8-bit
<i>MDMA_SOURCE_INC REASE_16BIT</i>	source address is increased by 16-bit
<i>MDMA_SOURCE_INC REASE_32BIT</i>	source address is increased by 32-bit
<i>MDMA_SOURCE_INC REASE_64BIT</i>	source address is increased by 64-bit
<i>MDMA_SOURCE_DEC REASE_8BIT</i>	source address is decreased by 8-bit
<i>MDMA_SOURCE_DEC REASE_16BIT</i>	source address is decreased by 16-bit
<i>MDMA_SOURCE_DEC REASE_32BIT</i>	source address is decreased by 32-bit
<i>MDMA_SOURCE_DEC REASE_64BIT</i>	source address is decreased by 32-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure source adress increment mode */
```

```
mdma_source_increment_config (MDMA_CH0, MDMA_SOURCE_INCREASE_16BIT);
```

### mdma\_destination\_increment\_config

The description of mdma\_destination\_increment\_config is shown as below:

**Table 3-747. Function mdma\_destination\_increment\_config**

<b>Function name</b>	mdma_destination_increment_config
<b>Function prototype</b>	void mdma_destination_increment_config(mdma_channel_enum channelx, uint32_t dinc);
<b>Function descriptions</b>	configure destination adress increment mode
<b>Precondition</b>	-
<b>The called functions</b>	-



Input parameter{in}	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum <i>mdma_channel_enum</i></a> .
Input parameter{in}	
<b>dinc</b>	destination address increment mode
<i>MDMA_DESTINATION_INCREASE_DISABLE</i>	no increment
<i>MDMA_DESTINATION_INCREASE_8BIT</i>	destination address is increased by 8-bit
<i>MDMA_DESTINATION_INCREASE_16BIT</i>	destination address is increased by 16-bit
<i>MDMA_DESTINATION_INCREASE_32BIT</i>	destination address is increased by 32-bit
<i>MDMA_DESTINATION_INCREASE_64BIT</i>	destination address is increased by 64-bit
<i>MDMA_DESTINATION_INCREASE_64BIT</i>	destination address is decreased by 8-bit
<i>MDMA_DESTINATION_DECREASE_16BIT</i>	destination address is decreased by 16-bit
<i>MDMA_DESTINATION_DECREASE_32BIT</i>	destination address is decreased by 32-bit
<i>MDMA_DESTINATION_DECREASE_64BIT</i>	destination address is decreased by 64-bit
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure source adress increment mode */
```

```
mdma_destination_increment_config (MDMA_CH0, MDMA_DESTINATION_INCREASE_16BIT);
```

### mdma\_channel\_bufferable\_write\_enable

The description of mdma\_channel\_bufferable\_write\_enable is shown as below:

**Table 3-748. Function mdma\_channel\_bufferable\_write\_enable**

<b>Function name</b>	mdma_channel_bufferable_write_enable
<b>Function prototype</b>	void mdma_channel_bufferable_write_enable(mdma_channel_enum channelx);
<b>Function descriptions</b>	enable MDMA channel bufferable write mode
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable MDMA channel bufferable write mode */
```

```
mdma_channel_bufferable_write_enable(MDMA_CH0);
```

### mdma\_channel\_bufferable\_write\_disable

The description of mdma\_channel\_bufferable\_write\_disable is shown as below:

**Table 3-749. Function mdma\_channel\_bufferable\_write\_disable**

Function name	mdma_channel_bufferable_write_disable
Function prototype	void mdma_channel_bufferable_write_disable(mdma_channel_enum channelx);
Function descriptions	disable MDMA channel bufferable write mode
Precondition	-
The called functions	-
Input parameter{in}	
channelx	MDMA channel
MDMA_CHx(x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable MDMA channel bufferable write mode */
```

```
mdma_channel_bufferable_write_disable(MDMA_CH0);
```

### mdma\_channel\_software\_request\_enable

The description of mdma\_channel\_software\_request\_enable is shown as below:

**Table 3-750. Function mdma\_channel\_software\_request\_enable**

Function name	mdma_channel_software_request_enable
---------------	--------------------------------------

<b>Function prototype</b>	void mdma_channel_software_request_enable(mdma_channel_enum channelx);
<b>Function descriptions</b>	enable MDMA channel software request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable MDMA channel software request */
mdma_channel_software_request_enable(MDMA_CH0);
```

### mdma\_channel\_enable

The description of mdma\_channel\_enable is shown as below:

**Table 3-751. Function mdma\_channel\_enable**

<b>Function name</b>	mdma_channel_enable
<b>Function prototype</b>	void mdma_channel_enable(mdma_channel_enum channelx);
<b>Function descriptions</b>	enable MDMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable MDMA channel */
mdma_channel_enable(MDMA_CH0);
```

### mdma\_channel\_disable

The description of mdma\_channel\_disable is shown as below:

**Table 3-752. Function mdma\_channel\_disable**

<b>Function name</b>	mdma_channel_disable
<b>Function prototype</b>	void mdma_channel_disable(mdma_channel_enum channelx);
<b>Function descriptions</b>	disable MDMA channel
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable MDMA channel */
```

```
mdma_channel_disable(MDMA_CH0);
```

### mdma\_transfer\_error\_direction\_get

The description of mdma\_transfer\_error\_direction\_get is shown as below:

**Table 3-753. Function mdma\_transfer\_error\_direction\_get**

<b>Function name</b>	mdma_transfer_error_direction_get
<b>Function prototype</b>	uint32_t mdma_transfer_error_direction_get(mdma_channel_enum channelx);
<b>Function descriptions</b>	get MDMA transfer error direction
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	transfer error direction
<i>MDMA_READ_ERROR</i>	read access error
<i>MDMA_WRITE_ERROR</i>	write access error

Example:

```
/* get MDMA transfer error direction */
```

```
uint32_t dir;

dir = mdma_transfer_error_direction_get(MDMA_CH0);
```

### mdma\_transfer\_error\_address\_get

The description of mdma\_transfer\_error\_address\_get is shown as below:

**Table 3-754. Function mdma\_transfer\_error\_address\_get**

<b>Function name</b>	mdma_transfer_error_address_get
<b>Function prototype</b>	uint32_t mdma_transfer_error_address_get(mdma_channel_enum channelx);
<b>Function descriptions</b>	get MDMA transfer error address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	the low 7 bits of the transfer error address, 0x00000000-0x0000007F

Example:

```
/* get MDMA transfer error address */

uint32_t err_addr;

err_addr = mdma_transfer_error_address_get(MDMA_CH0);
```

### mdma\_flag\_get

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

**Table 3-755. Function mdma\_flag\_get**

<b>Function name</b>	mdma_flag_get
<b>Function prototype</b>	FlagStatus mdma_flag_get(mdma_channel_enum channelx, uint32_t flag);
<b>Function descriptions</b>	get MDMA flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>flag</b>	MDMA flag

<i>MDMA_FLAG_ERR</i>	transfer error flag
<i>MDMA_FLAG_CHTCF</i>	channel transfer complete flag
<i>MDMA_FLAG_MBTCF</i>	multi-block transfer complete flag
<i>MDMA_FLAG_BTCF</i>	block transfer complete flag
<i>MDMA_FLAG_TCF</i>	buffer transfer complete flag
<i>MDMA_FLAG_REQAF</i>	request active flag
<i>MDMA_FLAG_LDTER</i> <i>R</i>	link data transfer error flag in the last transfer of the channel
<i>MDMA_FLAG_MDTER</i> <i>R</i>	mask data error flag
<i>MDMA_FLAG_ASERR</i>	address and size error flag
<i>MDMA_FLAG_BZERR</i>	block size error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get MDMA flag */
```

```
FlagStatus status;
```

```
status = mdma_flag_get(MDMA_CH0, MDMA_FLAG_TCF);
```

### mdma\_flag\_clear

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

**Table 3-756. Function mdma\_flag\_clear**

<b>Function name</b>	mdma_flag_clear
<b>Function prototype</b>	void mdma_flag_clear(mdma_channel_enum channelx, uint32_t flag);
<b>Function descriptions</b>	clear MDMA flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>flag</b>	MDMA flag
<i>MDMA_FLAG_ERR</i>	transfer error flag
<i>MDMA_FLAG_CHTCF</i>	channel transfer complete flag
<i>MDMA_FLAG_MBTCF</i>	multi-block transfer complete flag
<i>MDMA_FLAG_BTCF</i>	block transfer complete flag
<i>MDMA_FLAG_TCF</i>	buffer transfer complete flag

<i>MDMA_FLAG_LDTER</i> R	link data transfer error flag in the last transfer of the channel
<i>MDMA_FLAG_MDTER</i> R	mask data error flag
<i>MDMA_FLAG_ASERR</i>	address and size error flag
<i>MDMA_FLAG_BZERR</i>	block size error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear MDMA flag */
```

```
mdma_flag_clear(MDMA_CH0, MDMA_FLAG_TCF);
```

### mdma\_interrupt\_enable

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

**Table 3-757. Function mdma\_interrupt\_enable**

<b>Function name</b>	mdma_interrupt_enable
<b>Function prototype</b>	void mdma_interrupt_enable(mdma_channel_enum channelx, uint32_t interrupt);
<b>Function descriptions</b>	enable MDMA interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx</i> (x=0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Output parameter{out}</b>	
<b>interrupt</b>	MDMA interrupt
<i>MDMA_INT_ERR</i>	transfer error interrupt
<i>MDMA_INT_ERR</i>	channel transfer complete interrupt
<i>MDMA_INT_MBTC</i>	multi-block transfer complete interrupt
<i>MDMA_INT_BTC</i>	block transfer complete interrupt
<i>MDMA_INT_TC</i>	buffer transfer complete interrupt
<b>Return value</b>	
-	-

Example:

```
/* enable MDMA interrupt */
```

```
mdma_interrupt_enable(MDMA_CH0, MDMA_INT_TC);
```

## mdma\_interrupt\_disable

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

**Table 3-758. Function mdma\_interrupt\_disable**

<b>Function name</b>	mdma_interrupt_disable
<b>Function prototype</b>	void mdma_interrupt_disable(mdma_channel_enum channelx, uint32_t interrupt);
<b>Function descriptions</b>	disable MDMA interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx(x=0..15)</i>	MDMA channel selection, refer to <a href="#">Table 3-723. Enum mdma_channel_enum</a> .
<b>Input parameter{in}</b>	
<b>interrupt</b>	MDMA interrupt
<i>MDMA_INT_ERR</i>	transfer error interrupt
<i>MDMA_INT_ERR</i>	channel transfer complete interrupt
<i>MDMA_INT_MBTC</i>	multi-block transfer complete interrupt
<i>MDMA_INT_BTC</i>	block transfer complete interrupt
<i>MDMA_INT_TC</i>	buffer transfer complete interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable MDMA interrupt */
mdma_interrupt_disable(MDMA_CH0, MDMA_INT_TC);
```

## mdma\_interrupt\_flag\_get

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

**Table 3-759. Function mdma\_interrupt\_flag\_get**

<b>Function name</b>	mdma_interrupt_flag_get
<b>Function prototype</b>	FlagStatus mdma_interrupt_flag_get(mdma_channel_enum channelx, uint32_t int_flag);
<b>Function descriptions</b>	get MDMA interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel



<i>MDMA_CHx</i> ( <i>x</i> =0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum <i>mdma_channel_enum</i></a> .
<b>Input parameter{in}</b>	
<b>int_flag</b>	MDMA interrupt flag
<i>MDMA_INT_FLAG_ER</i> <i>R</i>	transfer error interrupt flag
<i>MDMA_INT_FLAG_CH</i> <i>TCF</i>	channel transfer complete interrupt flag
<i>MDMA_INT_FLAG_MB</i> <i>TCF</i>	multi-block transfer complete interrupt flag
<i>MDMA_INT_FLAG_BT</i> <i>CF</i>	block transfer complete interrupt flag
<i>MDMA_INT_FLAG_TC</i> <i>F</i>	buffer transfer complete interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get MDMA interrupt flag */
```

```
FlagStatus status;
```

```
status = mdma_interrupt_flag_get(MDMA_CH0, MDMA_INT_FLAG_TCF);
```

### mdma\_interrupt\_flag\_clear

The description of `mdma_interrupt_flag_clear` is shown as below:

**Table 3-760. Function `mdma_interrupt_flag_clear`**

<b>Function name</b>	<code>mdma_interrupt_flag_clear</code>
<b>Function prototype</b>	<code>void mdma_interrupt_flag_clear(mdma_channel_enum channelx, uint32_t int_flag);</code>
<b>Function descriptions</b>	clear MDMA interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>channelx</b>	MDMA channel
<i>MDMA_CHx</i> ( <i>x</i> =0..15)	MDMA channel selection, refer to <a href="#">Table 3-723. Enum <i>mdma_channel_enum</i></a> .
<b>Input parameter{in}</b>	
<b>int_flag</b>	MDMA interrupt flag
<i>MDMA_INT_FLAG_ER</i> <i>R</i>	transfer error interrupt flag

<i>MDMA_INT_FLAG_CH</i> <i>TCF</i>	channel transfer complete interrupt flag
<i>MDMA_INT_FLAG_MB</i> <i>TCF</i>	multi-block transfer complete interrupt flag
<i>MDMA_INT_FLAG_BT</i> <i>CF</i>	block transfer complete interrupt flag
<i>MDMA_INT_FLAG_TC</i> <i>F</i>	buffer transfer complete interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear MDMA interrupt flag */
```

```
mdma_interrupt_flag_clear(MDMA_CH0, MDMA_INT_FLAG_TCF);
```

## 3.23. OSPI

The OSPI is a specialized interface that communicate with external memories. The interface support single, dual, quad and octal SPI flash(PSRAMS, NAND, NOR flash, etc). The OSPI registers are listed in chapter [3.23.1](#), the OSPI firmware functions are introduced in chapter [3.23.2](#).

### 3.23.1. Descriptions of Peripheral registers

OSPI registers are listed in the table shown as below:

**Table 3-761. OSPI Registers**

Registers	Descriptions
OSPI_CTL	OSPI control register
OSPI_DCFG0	OSPI device configuration register
OSPI_DCFG1	OSPI device configuration register
OSPI_STAT	OSPI status register
OSPI_STATC	OSPI status clear register
OSPI_DTLEN	OSPI data length register
OSPI_ADDR	OSPI address register
OSPI_DATA	OSPI data register
OSPI_STATMK	OSPI status mask register
OSPI_STATMATCH	OSPI status match register
OSPI_INTERVAL	OSPI interval register
OSPI_TCFG	OSPI transfer configuration register
OSPI_TIMCFG	OSPI timing configuration register

Registers	Descriptions
OSPI_INS	OSPI instruction register
OSPI_ALTE	OSPI alternate bytes register
OSPI_WPTCFG	OSPI wrap transfer configuration register
OSPI_WPTIMCFG	OSPI wrap timing configuration register
OSPI_WPINS	OSPI wrap instruction register
OSPI_WPALTE	OSPI wrap alternate bytes register
OSPI_WTCFG	OSPI write transfer configuration register
OSPI_WTIMCFG	OSPI write timing configuration register
OSPI_WINS	OSPI write instruction register
OSPI_WALTE	OSPI write alternate bytes register

### 3.23.2. Descriptions of Peripheral functions

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

**Table 3-762. OSPI firmware function**

Function name	Function description
ospi_deinit	reset OSPI
ospi_struct_init	initialize the parameters of OSPI struct with the default values
ospi_init	initialize OSPI parameter
ospi_enable	enable OSPI
ospi_disable	disable OSPI
ospi_device_memory_type_config	configure device memory type
ospi_device_memory_size_config	configure device memory size
ospi_functional_mode_config	select functional mode
ospi_status_polling_config	configure status polling mode
ospi_status_mask_config	configure status mask
ospi_status_match_config	configure status match
ospi_interval_cycle_config	configure interval cycle
ospi_fifo_level_config	configure OSPI fifo threshold level
ospi_chip_select_high_cycle_config	configure chip select high cycle
ospi_prescaler_config	configure OSPI prescaler
ospi_dummy_cycles_config	configure dummy cycles number
ospi_delay_hold_cycle_config	configure delay hold 1/4 cycle
ospi_sample_shift_config	configure sample shift
ospi_data_length_config	configure data length
ospi_instruction_config	configure instruction mode
ospi_address_config	configure address mode
ospi_alternate_bytes_config	configure alternate bytes mode
ospi_data_config	configure data mode
ospi_data_transmit	OSPI transmit data
ospi_data_receive	OSPI receive data

Function name	Function description
ospi_dma_enable	enable OSPI DMA
ospi_dma_disable	disable OSPI DMA
ospi_wrap_size_config	configure wrap size
ospi_wrap_instruction_config	configure wrap instruction mode
ospi_wrap_address_config	configure wrap address mode
ospi_wrap_alternate_bytes_config	configure wrap alternate bytes mode
ospi_wrap_data_config	configure wrap data mode
ospi_wrap_dummy_cycles_config	configure wrap dummy cycles number
ospi_wrap_delay_hold_cycle_config	delay hold 1/4 cycle in wrap
ospi_wrap_sample_shift_config	configure sample shift in wrap
ospi_write_instruction_config	configure write instruction mode
ospi_write_address_config	configure write address mode
ospi_write_alternate_bytes_config	configure write alternate bytes mode
ospi_write_data_config	configure write data mode
ospi_write_dummy_cycles_config	configure write dummy cycles number
ospi_command_config	configure OSPI regular command parameter
ospi_transmit	transmit data
ospi_receive	receive data
ospi_autopolling_mode	configure the OSPI automatic polling mode
ospi_interrupt_enable	enable OSPI interrupt
ospi_interrupt_disable	disable OSPI interrupt
ospi_fifo_level_get	get OSPI fifo level
ospi_flag_get	get OSPI flag status
ospi_flag_clear	clear OSPI flag status
ospi_interrupt_flag_get	get OSPI interrupt status
ospi_interrupt_flag_clear	clear OSPI interrupt flag status

## Structure ospi\_parameter\_struct

**Table3-763. Structure ospi\_parameter\_struct**

Member name	Function description
prescaler	the prescaler factor for generating clock based on the kernel clock (0-255)
fifo_threshold	the threshold number of bytes in the FIFO (0-31)
sample_shift	specifies the sample shift (OSPI_SAMPLE_SHIFTING_NONE, OSPI_SAMPLE_SHIFTING_HALF_CYCLE)
device_size	specifies the device size (OSPI_MESZ_x_BYTES(x = 2, 4, 8, ..., 512, 1024), OSPI_MESZ_x_KBS(x = 2, 4, 8, ..., 512, 1024), OSPI_MESZ_x_MBS(x = 2, 4, 8, ..., 2048, 4096))
cs_hightime	the chip select high time (OSPI_CS_HIGH_TIME_x_CYCLE (x = 1, 2, ..., 63, 64))

memory_type	the external device type (OSPI_MICRON_MODE, OSPI_MACRONIX_MODE, OSPI_STANDARD_MODE, OSPI_MACRONIX_RAM_MODE)
wrap_size	indicates the wrap-size corresponding the external device configuration (OSPI_DIRECT, OSPI_WRAP_16BYTES, OSPI_WRAP_32BYTES, OSPI_WRAP_64BYTES, OSPI_WRAP_128BYTES)
delay_hold_cycle	allows to hold to 1/4 cycle the data (OSPI_DELAY_HOLD_NONE, OSPI_DELAY_HOLD_QUARTER_CYCLE)

### Structure ospi\_regular\_cmd\_struct

**Table3-764. Structure ospi\_regular\_cmd\_struct**

Member name	Function description
operation_type	indicates if the configuration applies to the common registers or to the registers for the write operation (OSPI_OPTYPE_COMMON_CFG, OSPI_OPTYPE_READ_CFG, OSPI_OPTYPE_WRITE_CFG, OSPI_OPTYPE_WRAP_CFG)
instruction	instruction to be sent (0-0xFFFFFFFF)
ins_mode	indicates the mode of the instruction (OSPI_INSTRUCTION_NONE, OSPI_INSTRUCTION_1_LINE, OSPI_INSTRUCTION_2_LINES, OSPI_INSTRUCTION_4_LINES, OSPI_INSTRUCTION_8_LINES)
ins_size	indicates the size of the instruction (OSPI_INSTRUCTION_8_BITS, OSPI_INSTRUCTION_16_BITS, OSPI_INSTRUCTION_24_BITS, OSPI_INSTRUCTION_32_BITS)
address	the address to be sent (0-0xFFFFFFFF)
addr_mode	the mode of the address (OSPI_ADDRESS_NONE, OSPI_ADDRESS_1_LINE, OSPI_ADDRESS_2_LINES, OSPI_ADDRESS_4_LINES, OSPI_ADDRESS_8_LINES)
addr_size	the size of the address (OSPI_ADDRESS_8_BITS, OSPI_ADDRESS_16_BITS, OSPI_ADDRESS_24_BITS, OSPI_ADDRESS_32_BITS)
addr_dtr_mode	enables or not the DTR mode for the address phase (OSPI_ADDRDTR_MODE_DISABLE, OSPI_ADDTR_MODE_ENABLE)
alter_bytes	the alternate bytes to be sent (0-0xFFFFFFFF)
alter_bytes_mode	the mode of the alternate bytes (OSPI_ALTERNATE_BYTES_NONE, OSPI_ALTERNATE_BYTES_1_LINE, OSPI_ALTERNATE_BYTES_2_LINES, OSPI_ALTERNATE_BYTES_4_LINES, OSPI_ALTERNATE_BYTES_8_LINES)

Member name	Function description
alter_bytes_size	the size of the alternate bytes (OSPI_ALTERNATE_BYTES_8_BITS, OSPI_ALTERNATE_BYTES_16_BITS, OSPI_ALTERNATE_BYTES_24_BITS, OSPI_ALTERNATE_BYTES_32_BITS)
alter_bytes_dtr_mode	enables or not the DTR mode for the alternate bytes phase (OSPI_ABDTR_MODE_DISABLE, OSPI_ABDTR_MODE_ENABLE)
data_mode	the mode of the data (OSPI_DATA_NONE, OSPI_DATA_1_LINE, OSPI_DATA_2_LINES, OSPI_DATA_4_LINES, OSPI_DATA_8_LINES)
nbdata	the number of data transferred (1-0xFFFFFFFF)
data_dtr_mode	enables or not the DTR mode for the data phase (OSPI_DADTR_MODE_DISABLE, OSPI_DADTR_MODE_ENABLE)
dummy_cycles	the number of dummy cycles inserted before data phase (OSPI_DUMYC_CYCLES_x (x = 0, 1, 2, ..., 30, 31))

### Structure ospi\_autopolling\_struct

**Table3-765. Structure ospi\_autopolling\_struct**

Member name	Function description
match	the value to be compared with the masked status register to get a match (0- 0xFFFFFFFF)
mask	the mask to be applied to the status bytes received (0- 0xFFFFFFFF)
interval	the number of clock cycles between two read during automatic polling phases (0- 0xFFFF)
match_mode	the method used for determining a match (OSPI_MATCH_MODE_AND, OSPI_MATCH_MODE_OR)
automatic_stop	specifies if automatic polling is stopped after a match (OSPI_AUTOMATIC_STOP_ABORT, OSPI_AUTOMATIC_STOP_MATCH)

### Enum ospi\_interrupt\_flag\_enum

**Table 3-766. Enum ospi\_interrupt\_flag\_enum**

Member name	Function description
OSPI_INT_FLAG_TERR	transfer error interrupt flag
OSPI_INT_FLAG_TC	transfer complete interrupt enable
OSPI_INT_FLAG_FT	fifo threshold interrupt flag
OSPI_INT_FLAG_SM	status match interrupt flag

### ospi\_deinit

The description of ospi\_deinit is shown as below:

**Table3-767. Function ospi\_deinit**

<b>Function name</b>	ospi_deinit
<b>Function prototype</b>	void ospi_deinit(uint32_t ospi_periph);
<b>Function descriptions</b>	reset the OSPI peripheral
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset the OSPI peripheral */
```

```
ospi_deinit();
```

### ospi\_struct\_init

The description of ospi\_struct\_init is shown as below:

**Table3-768. Function ospi\_struct\_init**

<b>Function name</b>	ospi_struct_init
<b>Function prototype</b>	void ospi_struct_init(ospi_parameter_struct *ospi_struct);
<b>Function descriptions</b>	initialize the parameters of OSPI struct with default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_struct</b>	OSPI parameter struct, the structure members can refer to members of the structure <a href="#">Table3-763. Structure ospi_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize the parameters of OSPI struct with default values */
```

```
ospi_parameter_struct ospi_struct;
```

```
ospi_struct_init(&ospi_struct);
```

### ospi\_init

The description of ospi\_init is shown as below:

**Table3-769. Function ospi\_init**

<b>Function name</b>	ospi_init
<b>Function prototype</b>	void ospi_init(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct);
<b>Function descriptions</b>	initialize OSPI parameter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>ospi_struct</b>	OSPI parameter struct, the structure members can refer to members of the structure <a href="#">Table3-763. Structure ospi_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* initialize OSPI parameter */

ospi_parameter_struct ospi_struct;

ospi_struct.prescaler = 8;

ospi_struct.fifo_threshold = OSPI_FIFO_THRESHOLD_10;

ospi_struct.sample_shift = OSPI_SAMPLE_SHIFTING_NONE;

ospi_struct.device_size = OSPI_MESZ_512_MBS;

ospi_struct.cs_hightime = OSPI_CS_HIGH_TIME_10_CYCLE;

ospi_struct.memory_type = OSPI_STANDARD_MODE;

ospi_struct.wrap_size = OSPI_DIRECT;

ospi_struct.delay_hold_cycle = OSPI_DELAY_HOLD_QUARTER_CYCLE;

osp_init(OSPI0, &ospi_struct);

```

### ospi\_enable

The description of ospi\_enable is shown as below:

**Table3-770. Function ospi\_enable**

<b>Function name</b>	ospi_enable
<b>Function prototype</b>	void ospi_enable(uint32_t ospi_periph);
<b>Function descriptions</b>	enable OSPI
<b>Precondition</b>	-
<b>The called functions</b>	-



Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable OSPI0 */
ospi_enable(OSPI0);
```

### ospi\_disable

The description of ospi\_disable is shown as below:

**Table3-771. Function ospi\_disable**

Function name	ospi_disable
Function prototype	void ospi_disable(uint32_t ospi_periph);
Function descriptions	disable OSPI
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable OSPI0 */
ospi_disable(OSPI0);
```

### ospi\_device\_memory\_type\_config

The description of ospi\_device\_memory\_type\_config is shown as below:

**Table3-772. Function ospi\_device\_memory\_type\_config**

Function name	ospi_device_memory_type_config
Function prototype	void ospi_device_memory_type_config(uint32_t ospi_periph, uint32_t dtysel);
Function descriptions	configure device memory type
Precondition	-
The called functions	-
Input parameter{in}	

<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>dtysel</b>	OSPI device type select
<i>OSPI_MICRON_MODE</i>	micron mode
<i>OSPI_MACRONIX_MODE</i>	micronix mode
<i>OSPI_STANDARD_MODE</i>	standard mode
<i>OSPI_MACRONIX_RAM_MODE</i>	micronix ram mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure device memory type */
```

```
ospi_device_memory_type_config(OSPI0, OSPI_STANDARD_MODE);
```

### ospi\_device\_memory\_size\_config

The description of `ospi_device_memory_size_config` is shown as below:

**Table3-773. Function `ospi_device_memory_size_config`**

<b>Function name</b>	<code>ospi_device_memory_size_config</code>
<b>Function prototype</b>	<code>void ospi_device_memory_size_config(uint32_t ospi_periph, uint32_t mesz);</code>
<b>Function descriptions</b>	configure device memory size
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>mesz</b>	device memory size
<i>OSPI_MESE_X_BYTES</i>	x = 2, 4, 8, ..., 512, 1024
<i>OSPI_MESE_X_KBS</i>	x = 2, 4, 8, ..., 512, 1024
<i>OSPI_MESE_X_MBS</i>	x = 2, 4, 8, ..., 2048, 4096
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure device memory size */
```

```
ospi_device_memory_size_config(OSPI0, OSPI_MESE_1024_MBS);
```

### ospi\_functional\_mode\_config

The description of ospi\_functional\_mode\_config is shown as below:

**Table3-774. Function ospi\_functional\_mode\_config**

<b>Function name</b>	ospi_functional_mode_config
<b>Function prototype</b>	void ospi_functional_mode_config(uint32_t ospi_periph, uint32_t fmod);
<b>Function descriptions</b>	select functional mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>fmod</b>	OSPI functional mode
OSPI_INDIRECT_WRITE	OSPI indirect write mode
OSPI_INDIRECT_READ	OSPI indirect read mode
OSPI_STATUS_POLLING	OSPI status polling mode
OSPI_MEMORY_MAPPED	OSPI memory mapped mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select functional mode */
```

```
ospi_functional_mode_config(OSPI0, OSPI_INDIRECT_WRITE);
```

### ospi\_status\_polling\_config

The description of ospi\_status\_polling\_config is shown as below:

**Table3-775. Function ospi\_status\_polling\_config**

<b>Function name</b>	ospi_status_polling_config
<b>Function prototype</b>	void ospi_status_polling_config(uint32_t ospi_periph, uint32_t stop, uint32_t mode);
<b>Function descriptions</b>	configure status polling mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>stop</b>	OSPI automatic stop
<i>OSPI_AUTOMATIC_STOP_MATCH</i>	status polling mode stop in match
<b>Input parameter{in}</b>	
<b>mode</b>	OSPI match mode
<i>OSPI_MATCH_MODE_AND</i>	status polling match mode and
<i>OSPI_MATCH_MODE_OR</i>	status polling match mode or
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure status polling mode */
```

```
ospi_status_polling_config(OSPI0, OSPI_AUTOMATIC_STOP_ABORT,
OSPI_MATCH_MODE_OR);
```

### ospi\_status\_mask\_config

The description of ospi\_status\_mask\_config is shown as below:

**Table3-776. Function ospi\_status\_mask\_config**

<b>Function name</b>	ospi_status_mask_config
<b>Function prototype</b>	void ospi_status_mask_config(uint32_t ospi_periph, uint32_t mask);
<b>Function descriptions</b>	configure status mask
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>mask</b>	status mask (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure status mask */
```

```
ospi_status_mask_config (OSPI0, 0x55555555);
```

## ospi\_status\_match\_config

The description of ospi\_status\_match\_config is shown as below:

**Table3-777. Function ospi\_status\_match\_config**

<b>Function name</b>	ospi_status_match_config
<b>Function prototype</b>	void ospi_status_match_config(uint32_t ospi_periph, uint32_t match);
<b>Function descriptions</b>	configure status match
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>match</b>	status match (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure status match */
ospi_status_match_config (OSPI0, 0x55555555);
```

## ospi\_interval\_cycle\_config

The description of ospi\_interval\_cycle\_config is shown as below:

**Table3-778. Function ospi\_interval\_cycle\_config**

<b>Function name</b>	ospi_interval_cycle_config
<b>Function prototype</b>	void ospi_interval_cycle_config(uint32_t ospi_periph, uint16_t interval);
<b>Function descriptions</b>	configure interval cycle
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>interval</b>	interval cycle (0-0xFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure interval cycle */
```

ospi\_interval\_cycle\_config (OSPI0, 0x5555);

### ospi\_fifo\_level\_config

The description of ospi\_fifo\_level\_config is shown as below:

**Table3-779. Function ospi\_fifo\_level\_config**

<b>Function name</b>	ospi_fifo_level_config
<b>Function prototype</b>	void ospi_fifo_level_config(uint32_t ospi_periph, uint32_t ftl)
<b>Function descriptions</b>	configure OSPI fifo threshold level
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>ftl</b>	FIFO threshold level
OSPI_FIFO_THRESHOLD_x	threshold level set to x (x = 1, 2, ..., 31, 32)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* ospi_fifo_level_config */
ospi_fifo_level_config(OSPI0, OSPI_FIFO_THRESHOLD_10);
```

### ospi\_chip\_select\_high\_cycle\_config

The description of ospi\_chip\_select\_high\_cycle\_config is shown as below:

**Table3-780. Function ospi\_chip\_select\_high\_cycle\_config**

<b>Function name</b>	ospi_chip_select_high_cycle_config
<b>Function prototype</b>	void ospi_chip_select_high_cycle_config(uint32_t ospi_periph, uint32_t cshc);
<b>Function descriptions</b>	configure chip select high cycle
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>cshc</b>	OSPI chip select high cycle
OSPI_CS_HIGH_TIME_x_CYCLE	chip select high cycle set to x (x = 1, 2, ..., 63, 64)
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure chip select high cycle */
```

```
ospi_chip_select_high_cycle_config(OSPI0, OSPI_CS_HIGH_TIME_3_CYCLE);
```

### ospi\_prescaler\_config

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

**Table3-781. Function ospi\_prescaler\_config**

<b>Function name</b>	ospi_prescaler_config
<b>Function prototype</b>	void ospi_prescaler_config(uint32_t ospi_periph, uint32_t psc);
<b>Function descriptions</b>	configure OSPI prescaler
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>psc</b>	the scaler factor for generating SCK based on the kernel clock (0-0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure OSPI prescaler */
```

```
ospi_prescaler_config(OSPI0, 0x08);
```

### ospi\_dummy\_cycles\_config

The description of ospi\_dummy\_cycles\_config is shown as below:

**Table3-782. Function ospi\_dummy\_cycles\_config**

<b>Function name</b>	ospi_dummy_cycles_config
<b>Function prototype</b>	void ospi_dummy_cycles_config(uint32_t ospi_periph, uint32_t dumyc);
<b>Function descriptions</b>	configure dummy cycles number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	

<b>dumyc</b>	number of dummy cycles
<i>OSPI_DUMYC_CYCLE</i> <i>S_x</i>	dummy cycles set to x (x = 0, 1, 2, ..., 30, 31)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure dummy cycles number */
```

```
ospi_dummy_cycles_config(OSPI0, OSPI_DUMYC_CYCLES_5);
```

### ospi\_delay\_hold\_cycle\_config

The description of `ospi_delay_hold_cycle_config` is shown as below:

**Table3-783. Function `ospi_delay_hold_cycle_config`**

<b>Function name</b>	<code>ospi_delay_hold_cycle_config</code>
<b>Function prototype</b>	<code>void ospi_delay_hold_cycle_config(uint32_t ospi_periph, uint32_t dehqc);</code>
<b>Function descriptions</b>	delay hold 1/4 cycle
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>dehqc</b>	OSPI delay hold quarter cycle
<i>OSPI_DELAY_HOLD_NONE</i>	OSPI no delay hold cycle
<i>OSPI_DELAY_HOLD_QUARTER_CYCLE</i>	OSPI delay hold 1/4 cycle
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* delay hold 1/4 cycle */
```

```
ospi_delay_hold_cycle_config(OSPI0, OSPI_DELAY_HOLD_QUARTER_CYCLE);
```

### ospi\_sample\_shift\_config

The description of `ospi_sample_shift_config` is shown as below:



**Table3-784. Function ospi\_sample\_shift\_config**

<b>Function name</b>	ospi_sample_shift_config
<b>Function prototype</b>	void ospi_sample_shift_config(uint32_t ospi_periph, uint32_t ssample);
<b>Function descriptions</b>	configure sample shift
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>ssample</b>	OSPI sample shift
OSPI_SAMPLE_SHIFTING_NONE	OSPI no sample shift
OSPI_SAMPLE_SHIFTING_HALF_CYCLE	OSPI have 1/2 cycle sample shift
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure sample shift */
```

```
ospi_sample_shift_config(OSPI0, OSPI_SAMPLE_SHIFTING_NONE);
```

### ospi\_data\_length\_config

The description of ospi\_data\_length\_config is shown as below:

**Table3-785. Function ospi\_data\_length\_config**

<b>Function name</b>	ospi_data_length_config
<b>Function prototype</b>	void ospi_data_length_config(uint32_t ospi_periph, uint32_t dtlen);
<b>Function descriptions</b>	configure data length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx (x=0,1)
<b>Input parameter{in}</b>	
<b>dtlen</b>	data length (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data length */
```

```
ospi_data_length_config(OSPI0, 0x00005555);
```

## ospi\_instruction\_config

The description of ospi\_instruction\_config is shown as below:

**Table3-786. Function ospi\_instruction\_config**

<b>Function name</b>	ospi_instruction_config
<b>Function prototype</b>	void ospi_instruction_config(uint32_t ospi_periph, uint32_t imod, uint32_t inssz, uint32_t instruction);
<b>Function descriptions</b>	configure instruction mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx (x=0,1)
<b>Input parameter{in}</b>	
<b>imod</b>	OSPI instruction mode
OSPI_INSTRUCTION_NONE	no instruction mode
OSPI_INSTRUCTION_1_LINE	instruction mode on a single line
OSPI_INSTRUCTION_2_LINES	instruction mode on two lines
OSPI_INSTRUCTION_4_LINES	instruction mode on four lines
OSPI_INSTRUCTION_8_LINES	instruction mode on eight lines
<b>Input parameter{in}</b>	
<b>inssz</b>	OSPI instruction size
OSPI_INSTRUCTION_8_BITS	8-bit instruction
OSPI_INSTRUCTION_16_BITS	16-bit instruction
OSPI_INSTRUCTION_24_BITS	24-bit instruction
OSPI_INSTRUCTION_32_BITS	32-bit instruction
<b>Input parameter{in}</b>	
<b>instruction</b>	the instruction to be sent (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure instruction mode */
```

```
ospi_instruction_config(OSPI0, OSPI_INSTRUCTION_8_LINES,
OSPI_INSTRUCTION_8_BITS, 0x00005555);
```

## ospi\_address\_config

The description of ospi\_address\_config is shown as below:

**Table3-787. Function ospi\_address\_config**

<b>Function name</b>	ospi_address_config
<b>Function prototype</b>	void ospi_address_config(uint32_t ospi_periph, uint32_t addrmod, uint32_t addrdrtr, uint32_t addrsz, uint32_t addr);
<b>Function descriptions</b>	configure address mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx (x=0,1)
<b>Input parameter{in}</b>	
<b>addrmod</b>	OSPI address mode
OSPI_ADDRESS_NONE	no address mode
OSPI_ADDRESS_1_LINE	address mode on a single line
OSPI_ADDRESS_2_LINES	address mode on two lines
OSPI_ADDRESS_4_LINES	address mode on four lines
OSPI_ADDRESS_8_LINES	address mode on eight lines
<b>Input parameter{in}</b>	
<b>addrdrtr</b>	OSPI address double transfer rate
OSPI_ADDRDTR_MODE_DISABLE	address double transfer rate mode disable
OSPI_ADDRDTR_MODE_ENABLE	address double transfer rate mode enable
<b>Input parameter{in}</b>	
<b>addrsz</b>	OSPI address size
OSPI_ADDRESS_8_BITS	address size on 8-bit address
OSPI_ADDRESS_16_BITS	address size on 16-bit address
OSPI_ADDRESS_24_BITS	address size on 24-bit address
OSPI_ADDRESS_32_BITS	address size on 32-bit address

<i>ITS</i>	
<b>Input parameter{in}</b>	
<b>addr</b>	the address to be sent (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure address mode */
```

```
ospi_address_config(OSPI0, OSPI_ADDRESS_8_LINES,
OSPI_ADDRDTR_MODE_DISABLE, OSPI_ADDRESS_32_BITS, 0xFFFFFFFF);
```

### ospi\_alterate\_bytes\_config

The description of `ospi_alterate_bytes_config` is shown as below:

**Table3-788. Function `ospi_alterate_bytes_config`**

<b>Function name</b>	<code>ospi_alterate_bytes_config</code>
<b>Function prototype</b>	<code>void ospi_alterate_bytes_config(uint32_t ospi_periph, uint32_t atlemod, uint32_t abdtr, uint32_t altesz, uint32_t alte)</code>
<b>Function descriptions</b>	configure alternate bytes mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx (x=0,1)
<b>Input parameter{in}</b>	
<b>atlemod</b>	OSPI alternate bytes mode
<code>OSPI_ALTERNATE_BYTES_NONE</code>	no alternate bytes mode
<code>OSPI_ALTERNATE_BYTES_1_LINE</code>	alternate mode on a single line
<code>OSPI_ALTERNATE_BYTES_2_LINES</code>	alternate bytes mode on two lines
<code>OSPI_ALTERNATE_BYTES_4_LINES</code>	alternate bytes mode on four lines
<code>OSPI_ALTERNATE_BYTES_8_LINES</code>	alternate bytes mode on eight lines
<b>Input parameter{in}</b>	
<b>abdtr</b>	OSPI alternate bytes double transfer rate
<code>OSPI_ABDTR_MODE_DISABLE</code>	alternate bytes double transfer rate mode disable
<code>OSPI_ABDTR_MODE_ENABLE</code>	alternate bytes double transfer rate mode enable

Input parameter{in}	
<b>altesz</b>	OSPI alternate bytes size
<i>OSPI_ALTERNATE_BYTES_8_BITS</i>	alternate bytes size on 8-bit address
<i>OSPI_ALTERNATE_BYTES_16_BITS</i>	alternate bytes size on 16-bit address
<i>OSPI_ALTERNATE_BYTES_24_BITS</i>	alternate bytes size on 24-bit address
<i>OSPI_ALTERNATE_BYTES_32_BITS</i>	alternate bytes size on 32-bit address
Input parameter{in}	
<b>alte</b>	The alternate bytes to be sent (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure alternate bytes mode */
```

```
ospi_alternate_bytes_config(OSPI0, OSPI_ALTERNATE_BYTES_8_LINES,
OSPI_ABDTR_MODE_DISABLE, OSPI_ALTERNATE_BYTES_32_BITS, 0xFFFFFFFF);
```

### ospi\_data\_config

The description of `ospi_data_config` is shown as below:

**Table3-789. Function `ospi_data_config`**

<b>Function name</b>	<code>ospi_data_config</code>
<b>Function prototype</b>	<code>void ospi_data_config(uint32_t ospi_periph, uint32_t datamod, uint32_t dadtr);</code>
<b>Function descriptions</b>	configure data mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>ospi_periph</b>	OSPIx(x=0,1)
Input parameter{in}	
<b>datamod</b>	OSPI data mode
<i>OSPI_DATA_NONE</i>	no data mode
<i>OSPI_DATA_1_LINE</i>	data mode on a single line
<i>OSPI_DATA_2_LINES</i>	data mode on two lines
<i>OSPI_DATA_4_LINES</i>	data mode on four lines
<i>OSPI_DATA_8_LINES</i>	data mode on eight lines
Input parameter{in}	
<b>dadtr</b>	OSPI data double transfer rate

<i>OSPI_DADTR_MODE_DISABLE</i>	data double transfer rate mode disable
<i>OSPI_DADTR_MODE_ENABLE</i>	data double transfer rate mode enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure data mode */
ospi_data_config(OSPI0, OSPI_DATA_8_LINES, OSPI_DADTR_MODE_ENABLE);
```

### ospi\_data\_transmit

The description of ospi\_data\_transmit is shown as below:

**Table3-790. Function ospi\_data\_transmit**

<b>Function name</b>	ospi_data_transmit
<b>Function prototype</b>	void ospi_data_transmit(uint32_t ospi_periph, uint32_t data);
<b>Function descriptions</b>	OSPI transmit data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx (x=0,1)
<b>Input parameter{in}</b>	
<b>data</b>	transmit data (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* OSPI transmit data */
ospi_data_transmit(OSPI0, 0x00FF0000);
```

### ospi\_data\_receive

The description of ospi\_data\_receive is shown as below:

**Table3-791. Function ospi\_data\_receive**

<b>Function name</b>	ospi_data_receive
<b>Function prototype</b>	uint32_t ospi_data_receive(uint32_t ospi_periph);
<b>Function descriptions</b>	OSPI receive data

Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* OSPI receive data */
ospi_data_receive(OSPI0);
```

### ospi\_dma\_enable

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

**Table3-792. Function ospi\_dma\_enable**

Function name	ospi_dma_enable
Function prototype	void ospi_dma_enable(uint32_t ospi_periph);
Function descriptions	enable OSPI DMA
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable OSPI DMA */
ospi_dma_enable(OSPI0);
```

### ospi\_dma\_disable

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

**Table3-793. Function ospi\_dma\_disable**

Function name	ospi_dma_disable
Function prototype	void ospi_dma_disable(uint32_t ospi_periph);
Function descriptions	disable OSPI DMA
Precondition	-
The called functions	-

Input parameter{in}	
<b>ospi_periph</b>	OSPIx(x=0,1)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable OSPI DMA */
ospi_dma_disable(OSPI0);
```

### ospi\_wrap\_size\_config

The description of ospi\_wrap\_size\_config is shown as below:

**Table3-794. Function ospi\_wrap\_size\_config**

<b>Function name</b>	ospi_wrap_size_config
<b>Function prototype</b>	void ospi_wrap_size_config(uint32_t ospi_periph, uint32_t wpsz);
<b>Function descriptions</b>	configure wrap size
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>ospi_periph</b>	OSPIx(x=0,1)
Input parameter{in}	
<b>wpsz</b>	OSPI wrap size set
<i>OSPI_DIRECT</i>	external memory indirect device does not support wrap read
<i>OSPI_WRAP_16BYTES</i>	external memory device supports wrap size of 16 bytes
<i>OSPI_WRAP_32BYTES</i>	external memory device supports wrap size of 32 bytes
<i>OSPI_WRAP_64BYTES</i>	external memory device supports wrap size of 64 bytes
<i>OSPI_WRAP_128BYTES</i>	external memory device supports wrap size of 128 bytes
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure wrap size */
ospi_wrap_size_config(OSPI0, OSPI_WRAP_32BYTES);
```



## ospi\_wrap\_instruction\_config

The description of ospi\_wrap\_instruction\_config is shown as below:

**Table3-795. Function ospi\_wrap\_instruction\_config**

<b>Function name</b>	ospi_wrap_instruction_config
<b>Function prototype</b>	void ospi_wrap_instruction_config(uint32_t ospi_periph, uint32_t imod, uint32_t inssz, uint32_t instruction);
<b>Function descriptions</b>	configure wrap instruction mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>imod</b>	OSPI instruction mode
OSPI_INSTRUCTION_NONE	no instruction mode
OSPI_INSTRUCTION_1_LINE	instruction mode on a single line
OSPI_INSTRUCTION_2_LINES	instruction mode on two lines
OSPI_INSTRUCTION_4_LINES	instruction mode on four lines
OSPI_INSTRUCTION_8_LINES	instruction mode on eight lines
<b>Input parameter{in}</b>	
<b>inssz</b>	OSPI instruction size
OSPI_INSTRUCTION_8_BITS	8-bit instruction
OSPI_INSTRUCTION_16_BITS	16-bit instruction
OSPI_INSTRUCTION_24_BITS	24-bit instruction
OSPI_INSTRUCTION_32_BITS	32-bit instruction
<b>Input parameter{in}</b>	
<b>instruction</b>	the instruction to be sent (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure wrap instruction mode */
```

```
ospi_wrap_instruction_config(OSPI0, OSPI_INSTRUCTION_8_LINES,
OSPI_INSTRUCTION_8_BITS, 0x00005555);
```

### ospi\_wrap\_address\_config

The description of ospi\_wrap\_address\_config is shown as below:

**Table3-796. Function ospi\_wrap\_address\_config**

<b>Function name</b>	ospi_wrap_address_config
<b>Function prototype</b>	void ospi_wrap_address_config (uint32_t ospi_periph, uint32_t addrmod, uint32_t addrdrtr, uint32_t addrsz, uint32_t addr);
<b>Function descriptions</b>	Configure wrap address mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx (x=0,1)
<b>Input parameter{in}</b>	
<b>addrmod</b>	OSPI address mode
OSPI_ADDRESS_NONE	no address mode
OSPI_ADDRESS_1_LINE	address mode on a single line
OSPI_ADDRESS_2_LINES	address mode on two lines
OSPI_ADDRESS_4_LINES	address mode on four lines
OSPI_ADDRESS_8_LINES	address mode on eight lines
<b>Input parameter{in}</b>	
<b>addrdrtr</b>	OSPI address double transfer rate
OSPI_ADDRDTR_MODE_DISABLE	address double transfer rate mode disable
OSPI_ADDRDTR_MODE_ENABLE	address double transfer rate mode enable
<b>Input parameter{in}</b>	
<b>addrsz</b>	OSPI address size
OSPI_ADDRESS_8_BITS	8-bit address
OSPI_ADDRESS_16_BITS	16-bit address
OSPI_ADDRESS_24_BITS	24-bit address
OSPI_ADDRESS_32_BITS	32-bit address

Input parameter{in}	
<b>addr</b>	the address to be sent (0-0xFFFFFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure wrap address mode */
```

```
ospi_wrap_address_config(OSPI0, OSPI_ADDRESS_8_LINES,
OSPI_ADDRDTR_MODE_DISABLE, OSPI_ADDRESS_32_BITS, 0xFFFFFFFF);
```

### ospi\_wrap\_alternate\_bytes\_config

The description of ospi\_wrap\_alternate\_bytes\_config is shown as below:

**Table3-797. Function ospi\_wrap\_alternate\_bytes\_config**

Function name	ospi_wrap_alternate_bytes_config
Function prototype	void ospi_wrap_alternate_bytes_config(uint32_t ospi_periph, uint32_t atlemod, uint32_t abdtr, uint32_t altesz, uint32_t alte)
Function descriptions	configure wrap alternate bytes mode
Precondition	-
The called functions	-
Input parameter{in}	
<b>ospi_periph</b>	OSPIx (x=0,1)
Input parameter{in}	
<b>atlemod</b>	OSPI alternate bytes mode
OSPI_ALTERNATE_BYTES_NONE	no alternate bytes mode
OSPI_ALTERNATE_BYTES_1_LINE	alternate mode on a single line
OSPI_ALTERNATE_BYTES_2_LINES	alternate bytes mode on two lines
OSPI_ALTERNATE_BYTES_4_LINES	alternate bytes mode on four lines
OSPI_ALTERNATE_BYTES_8_LINES	alternate bytes mode on eight lines
Input parameter{in}	
<b>abdtr</b>	OSPI alternate bytes double transfer rate
OSPI_ABDTR_MODE_DISABLE	alternate bytes double transfer rate mode disable
OSPI_ABDTR_MODE_ENABLE	alternate bytes double transfer rate mode enable
Input parameter{in}	

<b>altesz</b>	OSPI alternate bytes size
<i>OSPI_ALTERNATE_BYTES_8_BITS</i>	alternate bytes size on 8-bit address
<i>OSPI_ALTERNATE_BYTES_16_BITS</i>	alternate bytes size on 16-bit address
<i>OSPI_ALTERNATE_BYTES_24_BITS</i>	alternate bytes size on 24-bit address
<i>OSPI_ALTERNATE_BYTES_32_BITS</i>	alternate bytes size on 32-bit address
<b>Input parameter{in}</b>	
<b>alte</b>	The alternate bytes to be sent (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure wrap alternate bytes mode */
```

```
ospi_wrap_alternate_bytes_config(OSPI0, OSPI_ALTERNATE_BYTES_8_LINES,
OSPI_ABDTR_MODE_DISABLE, OSPI_ALTERNATE_BYTES_32_BITS, 0xFFFFFFFF);
```

### ospi\_wrap\_data\_config

The description of `ospi_wrap_data_config` is shown as below:

**Table3-798. Function `ospi_wrap_data_config`**

<b>Function name</b>	<code>ospi_wrap_data_config</code>
<b>Function prototype</b>	<code>void ospi_wrap_data_config(uint32_t ospi_periph, uint32_t datamod, uint32_t dadtr);</code>
<b>Function descriptions</b>	configure wrap data mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>datamod</b>	OSPI data mode
<i>OSPI_DATA_NONE</i>	no data mode
<i>OSPI_DATA_1_LINE</i>	data mode on a single line
<i>OSPI_DATA_2_LINES</i>	data mode on two lines
<i>OSPI_DATA_4_LINES</i>	data mode on four lines
<i>OSPI_DATA_8_LINES</i>	data mode on eight lines
<b>Input parameter{in}</b>	
<b>dadtr</b>	OSPI data double transfer rate
<i>OSPI_DADTR_MODE_</i>	data double transfer rate mode disable

<i>DISABLE</i>	
<i>OSPI_DADTR_MODE_ENABLE</i>	data double transfer rate mode enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure wrap data mode */
```

```
ospi_wrap_data_config(OSPI0, OSPI_DATA_8_LINES, OSPI_DADTR_MODE_ENABLE);
```

### ospi\_wrap\_dummy\_cycles\_config

The description of `ospi_wrap_dummy_cycles_config` is shown as below:

**Table3-799. Function `ospi_wrap_dummy_cycles_config`**

<b>Function name</b>	<code>ospi_wrap_dummy_cycles_config</code>
<b>Function prototype</b>	<code>void ospi_wrap_dummy_cycles_config(uint32_t ospi_periph, uint32_t dumyc);</code>
<b>Function descriptions</b>	configure wrap dummy cycles number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>dumyc</b>	dummy cycles number (0-0x1F)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure wrap dummy cycles number */
```

```
ospi_wrap_dummy_cycles_config(OSPI0, 0x0F);
```

### ospi\_wrap\_delay\_hold\_cycle\_config

The description of `ospi_wrap_delay_hold_cycle_config` is shown as below:

**Table3-800. Function `ospi_wrap_delay_hold_cycle_config`**

<b>Function name</b>	<code>ospi_wrap_delay_hold_cycle_config</code>
<b>Function prototype</b>	<code>void ospi_wrap_delay_hold_cycle_config(uint32_t ospi_periph, uint32_t dehq);</code>

<b>Function descriptions</b>	delay hold 1/4 cycle in wrap
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>dehqc</b>	OSPI delay hold quarter cycle
<b>OSPI_DELAY_HOLD_NONE</b>	OSPI no delay hold cycle
<b>OSPI_DELAY_HOLD_QUARTER_CYCLE</b>	OSPI delay hold 1/4 cycle
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* delay hold 1/4 cycle in wrap */
```

```
ospi_wrap_delay_hold_cycle_config(OSPI0, OSPI_DELAY_HOLD_QUARTER_CYCLE);
```

### ospi\_wrap\_sample\_shift\_config

The description of ospi\_wrap\_sample\_shift\_config is shown as below:

**Table3-801. Function ospi\_wrap\_sample\_shift\_config**

<b>Function name</b>	ospi_wrap_sample_shift_config
<b>Function prototype</b>	void ospi_wrap_sample_shift_config(uint32_t ospi_periph, uint32_t ssample);
<b>Function descriptions</b>	configure sample shift in wrap
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>ssample</b>	OSPI sample shift
<b>OSPI_SAMPLE_SHIFTING_NONE</b>	OSPI no sample shift
<b>OSPI_SAMPLE_SHIFTING_HALF_CYCLE</b>	OSPI have 1/2 cycle sample shift
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure sample shift in wrap */
```

```
ospi_wrap_sample_shift_config(OSPI0, OSPI_SAMPLE_SHIFTING_HALF_CYCLE);
```

## ospi\_write\_instruction\_config

The description of ospi\_write\_instruction\_config is shown as below:

**Table3-802. Function ospi\_write\_instruction\_config**

<b>Function name</b>	ospi_write_instruction_config
<b>Function prototype</b>	void ospi_write_instruction_config(uint32_t ospi_periph, uint32_t imod, uint32_t inssz, uint32_t instruction);
<b>Function descriptions</b>	configure write instruction mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>imod</b>	OSPI instruction mode
OSPI_INSTRUCTION_NONE	no instruction mode
OSPI_INSTRUCTION_1_LINE	instruction mode on a single line
OSPI_INSTRUCTION_2_LINES	instruction mode on two lines
OSPI_INSTRUCTION_4_LINES	instruction mode on four lines
OSPI_INSTRUCTION_8_LINES	instruction mode on eight lines
<b>Input parameter{in}</b>	
<b>inssz</b>	OSPI instruction size
OSPI_INSTRUCTION_8_BITS	instruction size on 8-bit address
OSPI_INSTRUCTION_16_BITS	instruction size on 16-bit address
OSPI_INSTRUCTION_24_BITS	instruction size on 24-bit address
OSPI_INSTRUCTION_32_BITS	instruction size on 32-bit address
<b>Input parameter{in}</b>	
<b>instruction</b>	the instruction to be sent (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* configure wrap instruction mode */
```

```
ospi_wrtie_instruction_config(OSPI0, OSPI_INSTRUCTION_8_LINES,
OSPI_INSTRUCTION_8_BITS, 0x00005555);
```

### ospi\_write\_address\_config

The description of ospi\_write\_address\_config is shown as below:

**Table3-803. Function ospi\_write\_address\_config**

Function name	ospi_write_address_config
Function prototype	void ospi_write_address_config(uint32_t ospi_periph, uint32_t addrm, uint32_t addrdtr, uint32_t addrsz, uint32_t addr);
Function descriptions	configure write address mode
Precondition	-
The called functions	-
Input parameter{in}	
ospi_periph	OSPIx (x=0,1)
Input parameter{in}	
addrmod	OSPI address mode
OSPI_ADDRESS_NONE	no address mode
OSPI_ADDRESS_1_LINE	address mode on a single line
OSPI_ADDRESS_2_LINES	address mode on two lines
OSPI_ADDRESS_4_LINES	address mode on four lines
OSPI_ADDRESS_8_LINES	address mode on eight lines
Input parameter{in}	
addrdtr	OSPI address double transfer rate
OSPI_ADDRDTR_MODE_DISABLE	address double transfer rate mode disable
OSPI_ADDRDTR_MODE_ENABLE	address double transfer rate mode enable
Input parameter{in}	
addrsz	OSPI address size
OSPI_ADDRESS_8_BITS	address size on 8-bit address
OSPI_ADDRESS_16_BITS	address size on 16-bit address



<i>ITS</i>	
<i>OSPI_ADDRESS_24_B</i>	address size on 24-bit address
<i>ITS</i>	
<i>OSPI_ADDRESS_32_B</i>	address size on 32-bit address
<i>ITS</i>	
<b>Input parameter{in}</b>	
<b>addr</b>	the address to be sent (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure write address mode */
```

```
ospi_write_address_config(OSPI0, OSPI_ADDRESS_8_LINES,
OSPI_ADDRDTR_MODE_DISABLE, OSPI_ADDRESS_32_BITS, 0xFFFFFFFF);
```

### ospi\_write\_alternate\_bytes\_config

The description of `ospi_write_alternate_bytes_config` is shown as below:

**Table3-804. Function `ospi_write_alternate_bytes_config`**

<b>Function name</b>	<code>ospi_write_alternate_bytes_config</code>
<b>Function prototype</b>	<code>void ospi_write_alternate_bytes_config(uint32_t ospi_periph, uint32_t atlemod, uint32_t abdtr, uint32_t altesz, uint32_t alte);</code>
<b>Function descriptions</b>	configure write alternate bytes mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx (x=0,1)
<b>Input parameter{in}</b>	
<b>atlemod</b>	OSPI alternate bytes mode
<i>OSPI_ALTERNATE_BYTES_NONE</i>	no alternate bytes mode
<i>OSPI_ALTERNATE_BYTES_1_LINE</i>	alternate mode on a single line
<i>OSPI_ALTERNATE_BYTES_2_LINES</i>	alternate bytes mode on two lines
<i>OSPI_ALTERNATE_BYTES_4_LINES</i>	alternate bytes mode on four lines
<i>OSPI_ALTERNATE_BYTES_8_LINES</i>	alternate bytes mode on eight lines
<b>Input parameter{in}</b>	
<b>abdtr</b>	OSPI alternate bytes double transfer rate

<i>OSPI_ABDTR_MODE_DISABLE</i>	alternate bytes double transfer rate mode disable
<i>OSPI_ABDTR_MODE_ENABLE</i>	alternate bytes double transfer rate mode enable
<b>Input parameter{in}</b>	
<b>altesz</b>	OSPI alternate bytes size
<i>OSPI_ALTERNATE_BYTES_8_BITS</i>	alternate bytes size on 8-bit address
<i>OSPI_ALTERNATE_BYTES_16_BITS</i>	alternate bytes size on 16-bit address
<i>OSPI_ALTERNATE_BYTES_24_BITS</i>	alternate bytes size on 24-bit address
<i>OSPI_ALTERNATE_BYTES_32_BITS</i>	alternate bytes size on 32-bit address
<b>Input parameter{in}</b>	
<b>alte</b>	The alternate bytes to be sent (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure write alternate bytes mode */
```

```
ospi_write_alternate_bytes_config(OSPI0, OSPI_ALTERNATE_BYTES_8_LINES,
OSPI_ABDTR_MODE_DISABLE, OSPI_ALTERNATE_BYTES_32_BITS, 0xFFFFFFFF);
```

### ospi\_write\_data\_config

The description of `ospi_write_data_config` is shown as below:

**Table3-805. Function `ospi_write_data_config`**

<b>Function name</b>	<code>ospi_write_data_config</code>
<b>Function prototype</b>	<code>void ospi_write_data_config(uint32_t ospi_periph, uint32_t datamod, uint32_t dadtr);</code>
<b>Function descriptions</b>	configure write data mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>datamod</b>	OSPI data mode
<i>OSPI_DATA_NONE</i>	no data mode
<i>OSPI_DATA_1_LINE</i>	data mode on a single line
<i>OSPI_DATA_2_LINES</i>	data mode on two lines

<i>OSPI_DATA_4_LINES</i>	data mode on four lines
<i>OSPI_DATA_8_LINES</i>	data mode on eight lines
<b>Input parameter{in}</b>	
<b><i>dadtr</i></b>	OSPI data double transfer rate
<i>OSPI_DADTR_MODE_DISABLE</i>	data double transfer rate mode disable
<i>OSPI_DADTR_MODE_ENABLE</i>	data double transfer rate mode enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure write data mode */
```

```
ospi_write_data_config(OSPI0, OSPI_DATA_8_LINES, OSPI_DADTR_MODE_ENABLE);
```

### ospi\_write\_dummy\_cycles\_config

The description of `ospi_write_dummy_cycles_config` is shown as below:

**Table3-806. Function `ospi_write_dummy_cycles_config`**

<b>Function name</b>	<code>ospi_write_dummy_cycles_config</code>
<b>Function prototype</b>	<code>void ospi_write_dummy_cycles_config(uint32_t ospi_periph, uint32_t dumyc);</code>
<b>Function descriptions</b>	configure write dummy cycles number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b><i>ospi_periph</i></b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b><i>dumyc</i></b>	number of dummy cycles
<i>OSPI_DUMYC_CYCLE_S_x</i>	dummy cycles set to x (x=0,1,2,...,30,31)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure write dummy cycles number */
```

```
ospi_write_dummy_cycles_config(OSPI0, OSPI_DUMYC_CYCLES_10);
```

## ospi\_command\_config

The description of ospi\_command\_config is shown as below:

**Table3-807. Function ospi\_write\_dummy\_cycles\_config**

<b>Function name</b>	ospi_write_dummy_cycles_config
<b>Function prototype</b>	void ospi_command_config(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, ospi_regular_cmd_struct *cmd_struct);
<b>Function descriptions</b>	configure OSPI regular command parameter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>ospi_struct</b>	OSPI parameter struct, the structure members can refer to members of the strcture <a href="#">Table3-763. Structure ospi_parameter_struct</a>
<b>Input parameter{in}</b>	
<b>cmd_struct</b>	OSPI command struct, the structure members can refer to members of the strcture <a href="#">Table3-764. Structure ospi_regular_cmd_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure OSPI regular command parameter */
ospi_command_config(OSPI0, ospi_struct, cmd_struct);
```

## ospi\_transmit

The description of ospi\_transmit is shown as below:

**Table3-808. Function ospi\_transmit**

<b>Function name</b>	ospi_transmit
<b>Function prototype</b>	void ospi_transmit(uint32_t ospi_periph, uint8_t *pdata);
<b>Function descriptions</b>	transmit data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>pdata</b>	pointer to data buffer
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* transmit data */
```

```
ospi_transmit(OSPI0, pdata);
```

### ospi\_receive

The description of ospi\_receive is shown as below:

**Table3-809. Function ospi\_receive**

<b>Function name</b>	ospi_receive
<b>Function prototype</b>	void ospi_receive(uint32_t ospi_periph, uint8_t *pdata);
<b>Function descriptions</b>	receive data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>pdata</b>	pointer to data buffer
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* receive data */
```

```
ospi_receive(OSPI0, pdata);
```

### ospi\_autopolling\_mode

The description of ospi\_autopolling\_mode is shown as below:

**Table3-810. Function ospi\_autopolling\_mode**

<b>Function name</b>	ospi_autopolling_mode
<b>Function prototype</b>	void ospi_autopolling_mode(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, ospi_autopolling_struct *autopl_cfg_struct);
<b>Function descriptions</b>	configure the OSPI automatic polling mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	

<b>ospi_struct</b>	OSPI parameter struct, the structure members can refer to members of the structure <a href="#">Table3-763. Structure ospi_parameter_struct</a>
<b>Input parameter{in}</b>	
<b>ospi_autopolling_struct</b>	OSPI autopolling struct, the structure members can refer to members of the structure <a href="#">Table3-765. Structure ospi_autopolling_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the OSPI automatic polling mode */
```

```
ospi_autopolling_mode(OSPI0, ospi_struct, autopl_cfg_struct);
```

### ospi\_interrupt\_enable

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

**Table3-811. Function ospi\_interrupt\_enable**

<b>Function name</b>	ospi_interrupt_enable
<b>Function prototype</b>	void ospi_interrupt_enable(uint32_t ospi_periph, uint8_t interrupt);
<b>Function descriptions</b>	enable OSPI interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>interrupt</b>	OSPI interrupt
<i>OSPI_INT_TERR</i>	transfer error interrupt
<i>OSPI_INT_TC</i>	transfer complete interrupt
<i>OSPI_INT_FT</i>	fifo threshold interrupt
<i>OSPI_INT_SM</i>	status match interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable OSPI interrupt */
```

```
ospi_interrupt_enable(OSPI0, OSPI_INT_TERR);
```

### ospi\_interrupt\_disable

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

Table3-812. Function `ospi_interrupt_disable`

Function name	<code>ospi_interrupt_disable</code>
Function prototype	<code>void ospi_interrupt_disable(uint32_t ospi_periph, uint8_t interrupt);</code>
Function descriptions	disable OSPI interrupt
Precondition	-
The called functions	-
Input parameter{in}	
<code>ospi_periph</code>	OSPIx(x=0,1)
Input parameter{in}	
<code>interrupt</code>	OSPI interrupt
<code>OSPI_INT_TERR</code>	transfer error interrupt
<code>OSPI_INT_TC</code>	transfer complete interrupt
<code>OSPI_INT_FT</code>	fifo threshold interrupt
<code>OSPI_INT_SM</code>	status match interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable OSPI interrupt */
ospi_interrupt_disable(OSPI0, OSPI_INT_TERR);
```

### `ospi_fifo_level_get`

The description of `ospi_fifo_level_get` is shown as below:

Table3-813. Function `ospi_fifo_level_get`

Function name	<code>ospi_fifo_level_get</code>
Function prototype	<code>uint32_t ospi_fifo_level_get(uint32_t ospi_periph);</code>
Function descriptions	get OSPI fifo level
Precondition	-
The called functions	-
Input parameter{in}	
<code>ospi_periph</code>	OSPIx(x=0,1)
Output parameter{out}	
-	-
Return value	
<code>uint32_t</code>	0-0x3F

Example:

```
/* get OSPI fifo level */
ospi_fifo_level_get(OSPI0);
```

## ospi\_flag\_get

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

**Table3-814. Function ospi\_flag\_get**

<b>Function name</b>	ospi_flag_get
<b>Function prototype</b>	FlagStatus ospi_flag_get(uint32_t ospi_periph, uint32_t flag);
<b>Function descriptions</b>	get OSPI flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>flag</b>	OSPI flag status
OSPI_FLAG_TERR	transfer error flag
OSPI_FLAG_TC	transfer complete flag
OSPI_FLAG_FT	fifo threshold flag
OSPI_FLAG_SM	status match flag
OSPI_FLAG_BUSY	busy flag
<b>Output parameter{out}</b>	
<b>FlagStatus</b>	SET or RESET
<b>Return value</b>	
-	-

Example:

```
/* get OSPI flag status */
ospi_flag_get(OSPI0, OSPI_FLAG_BUSY);
```

## ospi\_flag\_clear

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

**Table3-815. Function ospi\_flag\_clear**

<b>Function name</b>	ospi_flag_clear
<b>Function prototype</b>	void ospi_flag_clear(uint32_t ospi_periph, uint32_t flag);
<b>Function descriptions</b>	clear OSPI flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>flag</b>	OSPI clear flag status
OSPI_FLAG_TERR	transfer error flag
OSPI_FLAG_TC	transfer complete flag



<i>OSPI_FLAG_SM</i>	status match flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear OSPI flag status */
```

```
ospi_flag_clear(OSPI0, OSPI_STATC_TERRC);
```

### ospi\_interrupt\_flag\_get

The description of `ospi_interrupt_flag_get` is shown as below:

**Table3-816. Function `ospi_interrupt_flag_get`**

<b>Function name</b>	<code>ospi_interrupt_flag_get</code>
<b>Function prototype</b>	<code>FlagStatus ospi_interrupt_flag_get(uint32_t ospi_periph, uint8_t int_flag);</code>
<b>Function descriptions</b>	get OSPI interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>int_flag</b>	OSPI interrupt flag status
<i>OSPI_INT_FLAG_TERR</i>	transfer error interrupt flag
<i>OSPI_INT_FLAG_TC</i>	transfer complete interrupt flag
<i>OSPI_INT_FLAG_FT</i>	fifo threshold interrupt flag
<i>OSPI_INT_FLAG_SM</i>	status match interrupt flag
<b>Output parameter{out}</b>	
<b>FlagStatus</b>	SET or RESET
<b>Return value</b>	
-	-

Example:

```
/* get OSPI interrupt flag status */
```

```
ospi_interrupt_flag_get(OSPI0, OSPI_INT_FLAG_TERR);
```

### ospi\_interrupt\_flag\_clear

The description of `ospi_interrupt_flag_clear` is shown as below:

**Table3-817. Function `ospi_interrupt_flag_clear`**

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

<b>Function prototype</b>	void ospi_interrupt_flag_clear(uint32_t ospi_periph, uint32_t int_flag);
<b>Function descriptions</b>	clear OSPI interrupt flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	OSPIx(x=0,1)
<b>Input parameter{in}</b>	
<b>int_flag</b>	clear OSPI interrupt flag status
OSPI_INT_FLAG_TER R	transfer error interrupt flag
OSPI_INT_FLAG_TC	transfer complete interrupt flag
OSPI_INT_FLAG_SM	status match interrupt flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear OSPI interrupt flag status */
```

```
ospi_interrupt_flag_clear(OSPI0, OSPI_STATC_TERRC);
```

## 3.24. OSPIM

OSPI supports OSPI pin assignment with full matrix before alternate function map. The OSPIM registers are listed in chapter [3.24.1](#), the OSPIM firmware functions are introduced in chapter [3.24.2](#).

### 3.24.1. Descriptions of Peripheral registers

OSPI registers are listed in the table shown as below:

**Table 3-818. OSPIM Registers**

Registers	Descriptions
OSPI_PCFCFG0	OSPI I/O manager port configuration register 0

### 3.24.2. Descriptions of Peripheral functions

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

**Table 3-819. OSPIM firmware function**

Function name	Function description
ospim_deinit	reset the OSPIM peripheral
ospim_port_sck_source_select	select source of SCK for port
ospim_port_csn_config	configure CSN for port

Function name	Function description
ospim_port_io3_0_config	configure IO[3:0] for port
ospim_port_io3_0_source_select	select source of IO[3:0] for port
ospim_port_io7_4_config	configure IO[7:4] for port
ospim_port_io7_4_source_select	select source of IO[7:4] for port

## ospim\_deinit

The description of ospim\_deinit is shown as below:

**Table3-820. Function ospim\_deinit**

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

Example:

```
/* reset the OSPIM peripheral */
ospim_deinit();
```

## ospim\_port\_sck\_config

The description of ospim\_port\_sck\_config is shown as below:

**Table3-821. Function ospim\_port\_sck\_config**

Function name	ospim_port_sck_config
Function prototype	void ospim_port_sck_config(uint32_t sckcfg);
Function descriptions	configure SCK for port
Precondition	-
The called functions	-
Input parameter{in}	
sckcfg	enable or disable SCK
OSPIM_PORT_SCK_DISABLE	disable SCK
OSPIM_PORT_SCK_ENABLE	enable SCK
Output parameter{out}	
-	-

Return value	
-	-

Example:

```
/* configurate SCK for port */
```

```
ospim_port_sck_config(OSPIM_PORT_SCK_ENABLE);
```

### ospim\_port\_csn\_config

The description of ospim\_port\_csn\_config is shown as below:

**Table3-822. Function ospim\_port\_csn\_config**

Function name	ospim_port_csn_config
Function prototype	void ospim_port_csn_config(uint32_t csconfig);
Function descriptions	configurate CSN for port
Precondition	-
The called functions	-
Input parameter{in}	
<b>csnconfig</b>	enable or disable CSN
<i>OSPIM_PORT_CSN_DISABLE</i>	disable CSN
<i>OSPIM_PORT_CSN_ENABLE</i>	enable CSN
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configurate CSN for port */
```

```
ospim_port_csn_config(OSPIM_PORT_CSN_ENABLE);
```

### ospim\_port\_io3\_0\_config

The description of ospim\_port\_io3\_0\_config is shown as below:

**Table3-823. Function ospim\_port\_io3\_0\_config**

Function name	ospim_port_io3_0_config
Function prototype	void ospim_port_io3_0_config(uint32_t ioconfig);
Function descriptions	configurate IO[3:0] for port
Precondition	-
The called functions	-
Input parameter{in}	
<b>ioconfig</b>	enable or disable IO[3:0]

<i>OSPIM_IO_LOW_DISA BLE</i>	disable IO[3:0]
<i>OSPIM_IO_LOW_ENA BLE</i>	enable IO[3:0]
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configurate IO[3:0] for port */
```

```
ospim_port_io3_0_config(OSPIM_IO_LOW_ENABLE);
```

### ospim\_port\_io3\_0\_source\_select

The description of ospim\_port\_io3\_0\_source\_select is shown as below:

**Table3-824. Function ospim\_port\_io3\_0\_source\_select**

<b>Function name</b>	ospim_port_io3_0_source_select
<b>Function prototype</b>	void ospim_port_io3_0_source_select(uint32_t io_source);
<b>Function descriptions</b>	select source of IO[3:0] for port
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>csn_source</b>	source of IO[3:0]
<i>OSPIM_SRCPLIO_OS PI0_IO_LOW</i>	select OSPI0_IO[3:0]
<i>OSPIM_SRCPLIO_OS PI0_IO_HIGH</i>	select OSPI0_IO[7:4]
<i>OSPIM_SRCPLIO_OS PI1_IO_LOW</i>	select OSPI1_IO[3:0]
<i>OSPIM_SRCPLIO_OS PI1_IO_HIGH</i>	select OSPI1_IO[7:4]
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select source of IO[3:0] for port */
```

```
ospim_port_io3_0_source_select(OSPIM_SRCPLIO_OSPI0_IO_LOW);
```

## ospim\_port\_io7\_4\_config

The description of ospim\_port\_io7\_4\_config is shown as below:

**Table3-825. Function ospim\_port\_io7\_4\_config**

<b>Function name</b>	ospim_port_io7_4_config
<b>Function prototype</b>	void ospim_port_io7_4_config(uint32_t ioconfig);
<b>Function descriptions</b>	configure IO[7:4] for port
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ioconfig</b>	enable or disable IO[7:4]
OSPIM_IO_HIGH_DISABLE	disable IO[7:4]
OSPIM_IO_HIGH_ENABLE	enable IO[7:4]
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure IO[7:4] for port */
```

```
ospim_port_io7_4_config(OSPIM_IO_HIGH_ENABLE);
```

## ospim\_port\_io7\_4\_source\_select

The description of ospim\_port\_io7\_4\_source\_select is shown as below:

**Table3-826. Function ospim\_port\_io7\_4\_source\_select**

<b>Function name</b>	ospim_port_io7_4_source_select
<b>Function prototype</b>	void ospim_port_io7_4_source_select(uint32_t io_source);
<b>Function descriptions</b>	select source of IO[7:4] for port
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>csn_source</b>	source of IO[3:0]
OSPIM_SRCPHIO_OSPI0_IO_LOW	select OSPI0_IO[3:0]
OSPIM_SRCPHIO_OSPI0_IO_HIGH	select OSPI0_IO[7:4]
OSPIM_SRCPHIO_OSPI1_IO_LOW	select OSPI1_IO[3:0]
OSPIM_SRCPHIO_OSPI1_IO_HIGH	select OSPI1_IO[7:4]

<i>PI1_IO_HIGH</i>	
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select source of IO[7:4] for port */
```

```
ospim_port_io7_4_source_select(OSPIM_SRCPHIO_OSPI0_IO_LOW);
```

## 3.25. MISC

MISC is a software package that provide the interfaces for NVIC and SysTick. The NVIC and SysTick registers are listed in chapter [3.25.1](#), the MISC firmware functions are introduced in chapter [3.25.2](#).

### 3.25.1. Descriptions of Peripheral registers

**Table 3-827. 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
IP <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
CFSR <sup>(2)</sup>	Configurable Fault Status Register
HFSR <sup>(2)</sup>	HardFault Status Register
DFSR <sup>(2)</sup>	Debug Fault Status Register
MMFAR <sup>(2)</sup>	MemManage Fault Address Register
BFAR <sup>(2)</sup>	BusFault Address Register
AFSR <sup>(2)</sup>	Auxiliary Fault Status Register
ID_PFR <sup>(2)</sup>	Processor Feature Register

Registers	Descriptions
ID_DFR <sup>(2)</sup>	Debug Feature Register
ID_AFR <sup>(2)</sup>	Auxiliary Feature Register
ID_MFR <sup>(2)</sup>	Memory Model Feature Register
ID_ISAR <sup>(2)</sup>	Instruction Set Attributes Register
CLIDR <sup>(2)</sup>	Cache Level ID register
CTR <sup>(2)</sup>	Cache Type register
CCSIDR <sup>(2)</sup>	Cache Size ID Register
CSSELR <sup>(2)</sup>	Cache Size Selection Register
CPACR <sup>(2)</sup>	Coprocessor Access Control Register
STIR <sup>(2)</sup>	Software Triggered Interrupt Register
MVFR0 <sup>(2)</sup>	Media and VFP Feature Register 0
MVFR1 <sup>(2)</sup>	Media and VFP Feature Register 1
MVFR2 <sup>(2)</sup>	Media and VFP Feature Register 2
ICIALLU <sup>(2)</sup>	I-Cache Invalidate All to PoU
ICIMVAU <sup>(2)</sup>	I-Cache Invalidate by MVA to PoU
DCIMVAC <sup>(2)</sup>	D-Cache Invalidate by MVA to PoC
DCISW <sup>(2)</sup>	D-Cache Invalidate by Set-way
DCCMVAU <sup>(2)</sup>	D-Cache Clean by MVA to PoU
DCCMVAC <sup>(2)</sup>	D-Cache Clean by MVA to PoC
DCCSW <sup>(2)</sup>	D-Cache Clean by Set-way
DCCIMVAC <sup>(2)</sup>	D-Cache Clean and Invalidate by MVA to PoC
DCCISW <sup>(2)</sup>	D-Cache Clean and Invalidate by Set-way
ITCMCR <sup>(2)</sup>	Instruction Tightly-Coupled Memory Control Register
DTCMCR <sup>(2)</sup>	Data Tightly-Coupled Memory Control Registers
AHBPCR <sup>(2)</sup>	AHBP Control Register
CACR <sup>(2)</sup>	L1 Cache Control Register
AHBSCR <sup>(2)</sup>	AHB Slave Control Register
ABFSR <sup>(2)</sup>	Auxiliary Bus Fault Status Register

1. refer to the structure NVIC\_Type, is defined in the core\_cm7.h file

2. refer to the structure SCB\_Type, is defined in the core\_cm7.h file

**Table 3-828. 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\_cm7.h file

### 3.25.2. Descriptions of Peripheral functions

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



**Table 3-829. MISC firmware function**

Function name	Function description
nvic_priority_group_set	set the priority group
nvic_irq_enable	enable NVIC request
nvic_irq_disable	disable NVIC request
nvic_vector_table_set	set the NVIC vector table address
system_lowpower_set	set the state of the low power mode
system_lowpower_reset	reset the state of the low power mode
systick_clksource_set	set the systick clock source
mpu_region_struct_para_init	initialize mpu_region_init_struct with the default values
mpu_region_config	configure the MPU region
mpu_region_enable	enable MPU region

### Structure mpu\_region\_init\_struct

**Table 3-830. Structure mpu\_region\_init\_struct**

Member name	Function description
region_base_address	region base address
region_number	region number
region_size	region size
subregion_disable	subregion disable
tex_type	tex type
access_permission	access permissions(AP) field
access_shareable	shareable
access_cacheable	cacheable
access_bufferable	bufferable
instruction_exec	execute never

### Enum IRQn\_Type

**Table 3-831. Enum IRQn\_Type**

Member name	Function description
WWDGT_IRQn	WWDGT interrupt
VAVD_LVD_VOVD_IRQn	AVD/LVD/OVD through EXTI Line detection interrupt
TAMPER_STAMP_LSE_IRQn	RTC Tamper and TimeStamp from EXTI Interrupt, LXTAL clock stuck interrupt
RTC_WKUP_IRQn	RTC Wakeup from EXTI interrupt
FMC_IRQn	FMC global interrupt
RCU_IRQn	RCU global interrupt
EXTI0_IRQn	EXTI Line0 interrupt
EXTI1_IRQn	EXTI Line1 interrupt
EXTI2_IRQn	EXTI Line2 interrupt
EXTI3_IRQn	EXTI Line3 interrupt
EXTI4_IRQn	EXTI Line4 interrupt

Member name	Function description
DMA0_Channel0_IRQn	DMA0 Channel0 global interrupt
DMA0_Channel1_IRQn	DMA0 Channel1 global interrupt
DMA0_Channel2_IRQn	DMA0 Channel2 global interrupt
DMA0_Channel3_IRQn	DMA0 Channel3 global interrupt
DMA0_Channel4_IRQn	DMA0 Channel4 global interrupt
DMA0_Channel5_IRQn	DMA0 Channel5 global interrupt
DMA0_Channel6_IRQn	DMA0 Channel6 global interrupt
ADC0_1_IRQn	ADC0 and ADC1 global interrupt
EXTI5_9_IRQn	EXTI Line5-9 interrupt
TIMER0_BRK_IRQn	TIMER0 break interrupt
TIMER0_UP_IRQn	TIMER0 update interrupt
TIMER0_TRG_CMT_IRQn	TIMER0 trigger and commutation interrupt
TIMER0_Channel_IRQn	TIMER0 capture compare interrupt
TIMER1_IRQn	TIMER1 global interrupt
TIMER2_IRQn	TIMER2 global interrupt
TIMER3_IRQn	TIMER3 global interrupt
I2C0_EV_IRQn	I2C0 event interrupt
I2C0_ER_IRQn	I2C0 error interrupt
I2C1_EV_IRQn	I2C1 event interrupt
I2C1_ER_IRQn	I2C1 error interrupt
SPI0_IRQn	SPI0 global interrupt
SPI1_IRQn	SPI1 global interrupt
USART0_IRQn	USART0 global and wakeup interrupt
USART1_IRQn	USART1 global and wakeup interrupt
USART2_IRQn	USART2 global and wakeup interrupt
EXTI10_15_IRQn	EXTI Line10-15 interrupt
RTC_Alarm_IRQn	RTC alarm from EXTI interrupt
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 capture compare interrupt
DMA0_Channel7_IRQn	DMA0 Channel7 global interrupt
EXMC_IRQn	EXMC global interrupt
TIMER4_IRQn	TIMER4 global interrupt
SPI2_IRQn	SPI2 global interrupt
UART3_IRQn	UART3 global interrupt
UART4_IRQn	UART4 global interrupt
TIMER5_DACOVN_IRQn	TIMER5 global interrupt and DAC1/DAC0 underrun error interrupt
TIMER6_IRQn	TIMER6 global interrupt
DMA1_Channel0_IRQn	DMA1 Channel 0 global interrupt
DMA1_Channel1_IRQn	DMA1 Channel 1 global interrupt

Member name	Function description
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
DMA1_Channel6_IRQn	DMA1 Channel 6 global interrupt
DMA1_Channel7_IRQn	DMA1 Channel 7 global interrupt
USART5_IRQn	USART5 global and wakeup interrupt
I2C2_EV_IRQn	I2C2 event interrupt
I2C2_ER_IRQn	I2C2 error interrupt
USBHS0_EP1_Out_IRQn	USBHS0 endpoint 1 out interrupt
USBHS0_EP1_In_IRQn	USBHS0 endpoint 1 in interrupt
USBHS0_WKUP_IRQn	USBHS0 wakeup from EXTI interrupt
USBHS0_IRQn	USBHS0 global interrupt
TRNG_IRQn	TRNG global interrupt
FPU_IRQn	FPU global interrupt
UART6_IRQn	UART6 global interrupt
UART7_IRQn	UART7 global interrupt
SPI3_IRQn	SPI3 global interrupt
SPI4_IRQn	SPI4 global interrupt
SPI5_IRQn	SPI5 global interrupt
QSPI0_IRQn	OSPI0 global interrupt
I2C3_EV_IRQn	I2C3 event interrupt
I2C3_ER_IRQn	I2C3 error interrupt
DMAMUX0_OVR_IRQn	DMAMUX overrun interrupt
HPDF0_IRQn	HPDF global interrupt 0
HPDF1_IRQn	HPDF global interrupt 1
HPDF2_IRQn	HPDF global interrupt 2
HPDF3_IRQn	HPDF global interrupt 3
TIMER14_IRQn	TIMER14 global interrupt
TIMER15_IRQn	TIMER15 global interrupt
TIMER16_IRQn	TIMER16 global interrupt
MDMA_IRQn	MDMA global interrupt
ADC2_IRQn	ADC2 global interrupt
CMP0_1_IRQn	CMP0 and CMP1 global interrupt, CMP0 and CMP1 through EXTI Line detection interrupt
WWDGT_RST_IRQn	WWDGT reset interrupt
CTC_IRQn	CTC interrupt
ECC_IRQn	RAMECCMU global interrupt
OSPI1_IRQn	OSPI1 global interrupt
FAC_IRQn	FAC global interrupt
TMU_IRQn	TMU global interrupt

Member name	Function description
TIMER22_IRQn	TIMER22 global interrupt
TIMER23_IRQn	TIMER23 global interrupt
TIMER40_IRQn	TIMER40 global interrupt
TIMER41_IRQn	TIMER41 global interrupt
TIMER42_IRQn	TIMER42 global interrupt
TIMER43_IRQn	TIMER43 global interrupt
TIMER44_IRQn	TIMER44 global interrupt
TIMER50_IRQn	TIMER50 global interrupt
TIMER51_IRQn	TIMER51 global interrupt
USBHS1_EP1_Out_IRQn	USBHS1 endpoint 1 out interrupt
USBHS1_EP1_In_IRQn	USBHS1 endpoint 1 in interrupt
USBHS1_WKUP_IRQn	USBHS1 wakeup from EXTI interrupt
USBHS1_IRQn	USBHS1 global interrupt
CAN0_WKUP_IRQn	CAN0 wakeup through EXTI Line detection interrupt
CAN0_MSGBUFFER_IRQn	CAN0 interrupt for message buffer
CAN0_BUSOFF_IRQn	CAN0 interrupt for Bus off / Bus off done
CAN0_ER_IRQn	CAN0 interrupt for Error
CAN0_ER_FAST_IRQn	CAN0 interrupt for Error in fast transmission
CAN0_TX_WARNING_IRQn	CAN0 interrupt for Transmit warning
CAN0_RX_WARNING_IRQn	CAN0 interrupt for Receive warning
CAN1_WKUP_IRQn	CAN1 wakeup through EXTI Line detection interrupt
CAN1_MSGBUFFER_IRQn	CAN1 interrupt for message buffer
CAN1_BUSOFF_IRQn	CAN1 interrupt for Bus off / Bus off done
CAN1_ER_IRQn	CAN1 interrupt for Error
CAN1_ER_FAST_IRQn	CAN1 interrupt for Error in fast transmission
CAN1_TX_WARNING_IRQn	CAN1 interrupt for Transmit warning
CAN1_RX_WARNING_IRQn	CAN1 interrupt for Receive warning
CAN2_WKUP_IRQn	CAN2 wakeup through EXTI Line detection interrupt
CAN2_MSGBUFFER_IRQn	CAN2 interrupt for message buffer
CAN2_BUSOFF_IRQn	CAN2 interrupt for Bus off / Bus off done
CAN2_ER_IRQn	CAN2 interrupt for Error
CAN2_ER_FAST_IRQn	CAN2 interrupt for Error in fast transmission
CAN2_TX_WARNING_IRQn	CAN2 interrupt for Transmit warning
CAN2_RX_WARNING_IRQn	CAN2 interrupt for Receive warning
EFUSE_IRQn	EFUSE global interrupt
I2C0_WKUP_IRQn	I2C0 wakeup through EXTI Line detection interrupt
I2C1_WKUP_IRQn	I2C1 wakeup through EXTI Line detection interrupt
I2C2_WKUP_IRQn	I2C2 wakeup through EXTI Line detection interrupt
I2C3_WKUP_IRQn	I2C3 wakeup through EXTI Line detection interrupt
LPDTS_IRQn	LPDTS interrupt
LPDTS_WKUP_IRQn	LPDTS wakeup through EXTI Line detection interrupt

Member name	Function description
TIMER0_DEC_IRQn	TIMER0 DEC interrupt
TIMER7_DEC_IRQn	TIMER7 DEC interrupt
TIMER1_DEC_IRQn	TIMER1 DEC interrupt
TIMER2_DEC_IRQn	TIMER2 DEC interrupt
TIMER3_DEC_IRQn	TIMER3 DEC interrupt
TIMER4_DEC_IRQn	TIMER4 DEC interrupt
TIMER22_DEC_IRQn	TIMER22 DEC interrupt
TIMER23_DEC_IRQn	TIMER23 DEC interrupt

### **nvic\_priority\_group\_set**

The description of nvic\_priority\_group\_set is shown as below:

**Table 3-832. Function nvic\_priority\_group\_set**

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

Example:

```
/* priority group configuration, 0 bits for pre-emption priority 4 bits for subpriority */
nvic_priority_group_set(NVIC_PRIGROUP_PRE0_SUB4);
```

### **nvic\_irq\_enable**

The description of nvic\_irq\_enable is shown as below:

Table 3-833. Function nvic\_irq\_enable

Function name	nvic_irq_enable
Function prototype	void nvic_irq_enable(IRQn_Type nvic_irq, uint8_t nvic_irq_pre_priority, uint8_t nvic_irq_sub_priority);
Function descriptions	enable NVIC interrupt request
Precondition	-
The called functions	nvic_priority_group_set / __NVIC_SetPriority
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to <a href="#">Table 3-831. Enum IRQn_Type</a>
Input parameter{in}	
nvic_irq_pre_priority	the pre-emption priority needed to set
Input parameter{in}	
nvic_irq_sub_priority	the subpriority needed to set
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### nvic\_irq\_disable

The description of nvic\_irq\_disable is shown as below:

Table 3-834. Function nvic\_irq\_disable

Function name	nvic_irq_disable
Function prototype	void nvic_irq_disable(IRQn_Type nvic_irq);
Function descriptions	disable NVIC interrupt request
Precondition	-
The called functions	NVIC_DisableIRQ
Input parameter{in}	
nvic_irq	NVIC interrupt, refer to enum <a href="#">Table 3-831. Enum IRQn_Type</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

## nvic\_vector\_table\_set

The description of nvic\_vector\_table\_set is shown as below:

**Table 3-835. 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>	-
<b>Input parameter{in}</b>	
<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>	
<b>Input parameter{in}</b>	
<b>offset</b>	vector table offset
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-836. 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
<i>SCB_LPM_SLEEP_EXIT_ISR</i>	if chose this para, the system always enter low power mode by exiting from ISR
<i>SCB_LPM_DEEPSLEEP</i>	if chose this para, the system will enter the DEEPSLEEP mode
<i>P</i>	
<i>SCB_LPM_WAKE_BY_ALL_INT</i>	if chose this para, the lowpower mode can be woke up by all the enable and disable interrupts
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

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

### system\_lowpower\_reset

The description of system\_lowpower\_reset is shown as below:

**Table 3-837. Function system\_lowpower\_reset**

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

Example:

```
/* the system will exit low power mode by exiting from ISR */
system_lowpower_reset(SCB_LPM_SLEEP_EXIT_ISR);
```

### systick\_clksource\_set

The description of systick\_clksource\_set is shown as below:

**Table 3-838. 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>	-



Input parameter{in}	
<b>systick_clksource</b>	the systick clock source needed to choose
<i>SYSTICK_CLKSOURC E_CKSYS</i>	systick clock source is from CK_SYS
<i>SYSTICK_CLKSOURC E_CKSYS_DIV8</i>	systick clock source is from CK_SYS / 8
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* systick clock source is CK_SYS / 8 */
systick_clksource_set(SYSTICK_CLKSOURCE_CKSYS_DIV8);
```

### mpu\_region\_struct\_para\_init

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

**Table 3-839. Function mpu\_region\_struct\_para\_init**

<b>Function name</b>	mpu_region_struct_para_init
<b>Function prototype</b>	void mpu_region_struct_para_init(mpu_region_init_struct *mpu_init_struct);
<b>Function descriptions</b>	initialize mpu_region_init_struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>mpu_init_struct</b>	MPU initialization structure, refer to <a href="#">Table 3-830. Structure mpu_region_init_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
mpu_region_init_struct mpu_init_para;

/* initialize mpu_region_init_struct with the default values */
mpu_region_struct_para_init(&mpu_init_para);
```

### mpu\_region\_config

The description of mpu\_region\_config is shown as below:

**Table 3-840. Function mpu\_region\_config**

<b>Function name</b>	mpu_region_config
----------------------	-------------------

<b>Function prototype</b>	void mpu_region_config(mpu_region_init_struct *mpu_init_struct);
<b>Function descriptions</b>	configure the MPU region
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mpu_init_struct</b>	MPU initialization structure, refer to <a href="#">Table 3-830. Structure mpu_region_init_struct.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
mpu_region_init_struct mpu_init_para;

mpu_region_struct_para_init(&mpu_init_para);

mpu_init_para.region_base_address = 0x24004000;

mpu_init_para.region_number = MPU_REGION_3;

mpu_init_para.access_permission = MPU_AP_PRIV_RW;

mpu_init_para.tex_type = MPU_TEX_TYPE1;

/* configure the MPU region */

mpu_region_config(&mpu_init_para);
```

## mpu\_region\_enable

The description of mpu\_region\_enable is shown as below:

**Table 3-841. Function mpu\_region\_enable**

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

Example:

```
/* enable MPU region */
```

```
mpu_region_enable();
```

## 3.26. 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 [3.26.1](#), the PMU firmware functions are introduced in chapter [3.26.2](#).

### 3.26.1. Descriptions of Peripheral registers

PMU registers are listed in the table shown as below:

**Table 3-842. PMU Registers**

Registers	Descriptions
PMU_CTL0	PMU control register 0
PMU_CS	PMU control and status register
PMU_CTL1	PMU control register 1
PMU_CTL2	PMU control register 2
PMU_CTL3	PMU control register 3
PMU_PAR	PMU parameter register

### 3.26.2. Descriptions of Peripheral functions

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

**Table 3-843. PMU firmware function**

Function name	Function description
pmu_deinit	reset PMU registers
pmu_lvd_select	select low voltage detector threshold
pmu_lvd_enable	enable PMU lvd
pmu_lvd_disable	disable PMU lvd
pmu_vavd_select	select analog voltage detector threshold
pmu_vavd_enable	enable PMU analog voltage detector
pmu_vavd_disable	disable PMU analog voltage detector
pmu_vovd_enable	enable PMU V <sub>0.9V</sub> core voltage detector
pmu_vovd_disable	disable PMU V <sub>0.9V</sub> core voltage detector
pmu_ldo_output_select	control the V <sub>0.9V</sub> core voltage level
pmu_sldo_output_select	Deep-sleep mode V <sub>0.9V</sub> core voltage select
pmu_vbat_charging_select	PMU V <sub>BAT</sub> battery charging resistor selection
pmu_vbat_charging_enable	enable V <sub>BAT</sub> battery charging
pmu_vbat_charging_disable	disable V <sub>BAT</sub> battery charging
pmu_vbat_temp_monitor_enable	enable V <sub>BAT</sub> and temperature monitoring
pmu_vbat_temp_monitor_disable	disable V <sub>BAT</sub> and temperature monitoring
pmu_usb_regulator_enable	enable USB regulator

Function name	Function description
pmu_usb_regulator_disable	disable USB regulator
pmu_usb_voltage_detector_enable	enable VDD33USB voltage level detector
pmu_usb_voltage_detector_disable	disable VDD33USB voltage level detector
pmu_smpps_ldo_supply_config	power supply configurations
pmu_to_sleepmode	enter sleep mode
pmu_to_deepsleepmode	enter deepsleep mode
pmu_to_standbymode	enter standby mode
pmu_wakeup_pin_enable	enable PMU wakeup pin
pmu_wakeup_pin_disable	disable PMU wakeup pin
pmu_backup_write_enable	enable backup domain write
pmu_backup_write_disable	disable backup domain write
pmu_backup_voltage_stabilizer_enable	enable backup voltage stabilizer
pmu_backup_voltage_stabilizer_disable	disable backup voltage stabilizer
pmu_enter_deepsleep_wait_time_config	configure IRC counter before enter Deep-sleep mode
pmu_exit_deepsleep_wait_time_config	configure IRC counter before exit Deep-sleep mode
pmu_flag_get	get flag state
pmu_flag_clear	clear flag bit

## pmu\_deinit

The description of pmu\_deinit is shown as below:

**Table 3-844. Function pmu\_deinit**

<b>Function name</b>	pmu_deinit
<b>Function prototype</b>	void pmu_deinit(void);
<b>Function descriptions</b>	reset PMU 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 PMU register */
pmu_deinit();
```

## pmu\_lvd\_select

The description of pmu\_lvd\_select is shown as below:

Table 3-845. Function pmu\_lvd\_select

Function name	pmu_lvd_select
Function prototype	void pmu_lvd_select(uint32_t lvd_t_n);
Function descriptions	select low voltage detector threshold
Precondition	-
The called functions	-
Input parameter{in}	
lvd_t_n	voltage threshold value
PMU_LVDT_0	voltage threshold is 2.1V
PMU_LVDT_1	voltage threshold is 2.3V
PMU_LVDT_2	voltage threshold is 2.4V
PMU_LVDT_3	voltage threshold is 2.6V
PMU_LVDT_4	voltage threshold is 2.7V
PMU_LVDT_5	voltage threshold is 2.9V
PMU_LVDT_6	voltage threshold is 3.0V
PMU_LVDT_7	input analog voltage on PB7 (compared with 0.8V)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select low voltage detector threshold as 3.0V */
```

```
pmu_lvd_select(PMU_LVDT_6);
```

### pmu\_lvd\_enable

The description of pmu\_lvd\_enable is shown as below:

Table 3-846. Function pmu\_lvd\_enable

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

Example:

```
/* enable PMU lvd */
```

```
pmu_lvd_enable();
```

## pmu\_lvd\_disable

The description of pmu\_lvd\_disable is shown as below:

**Table 3-847. Function pmu\_lvd\_disable**

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

Example:

```
/* disable PMU lvd */
```

```
pmu_lvd_disable();
```

## pmu\_vavd\_select

The description of pmu\_vavd\_select is shown as below:

**Table 3-848. Function pmu\_vavd\_select**

<b>Function name</b>	pmu_vavd_select
<b>Function prototype</b>	void pmu_vavd_select(uint32_t avdt_n);
<b>Function descriptions</b>	select analog voltage detector threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>avdt_n</b>	select analog voltage detector threshold
<i>PMU_VAVDVC_0</i>	voltage threshold of analog voltage detector is 1.7V
<i>PMU_VAVDVC_1</i>	voltage threshold of analog voltage detector is 2.1V
<i>PMU_VAVDVC_2</i>	voltage threshold of analog voltage detector is 2.5V
<i>PMU_VAVDVC_3</i>	voltage threshold of analog voltage detector is 2.8V
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select analog voltage detector threshold 2.8V */
```

```
pmu_vavd_select(PMU_VAVDVC_3);
```

### pmu\_vavd\_enable

The description of pmu\_vavd\_enable is shown as below:

**Table 3-849. Function pmu\_vavd\_enable**

<b>Function name</b>	pmu_vavd_enable
<b>Function prototype</b>	void pmu_vavd_enable(void);
<b>Function descriptions</b>	enable PMU analog voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable PMU analog voltage detector */
```

```
pmu_vavd_enable();
```

### pmu\_vavd\_disable

The description of pmu\_vavd\_disable is shown as below:

**Table 3-850. Function pmu\_vavd\_disable**

<b>Function name</b>	pmu_vavd_disable
<b>Function prototype</b>	void pmu_vavd_disable(void);
<b>Function descriptions</b>	disable PMU analog voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable PMU analog voltage detector */
```

```
pmu_vavd_disable();
```

## pmu\_vovd\_enable

The description of pmu\_vovd\_enable is shown as below:

**Table 3-851. Function pmu\_cvd\_enable**

<b>Function name</b>	pmu_vovd_enable
<b>Function prototype</b>	void pmu_vovd_enable(void);
<b>Function descriptions</b>	enable PMU V <sub>0.9V</sub> voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable PMU V0.9V voltage detector */
pmu_vovd_enable();
```

## pmu\_vovd\_disable

The description of pmu\_vovd\_disable is shown as below:

**Table 3-852. Function pmu\_vovd\_disable**

<b>Function name</b>	pmu_vovd_disable
<b>Function prototype</b>	void pmu_vovd_disable(void);
<b>Function descriptions</b>	disable PMU V <sub>0.9V</sub> core voltage detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable PMU V0.9V voltage detector */
pmu_cvd_disable();
```

## pmu\_ldo\_output\_select

The description of pmu\_ldo\_output\_select is shown as below:



Table 3-853. Function pmu\_ldo\_output\_select

Function name	pmu_ldo_output_select
Function prototype	void pmu_ldo_output_select(uint32_t ldo_n);
Function descriptions	control the V0.9V core voltage level
Precondition	-
The called functions	-
Input parameter{in}	
ldo_n	Select LDO output voltage
PMU_LDOVS_0	LDO output voltage 0.8V mode
PMU_LDOVS_1	LDO output voltage 0.85V mode
PMU_LDOVS_2	LDO output voltage 0.9V mode
PMU_LDOVS_3	LDO output voltage 0.95V mode
PMU_LDOVS_4	LDO output voltage 0.975V mode
PMU_LDOVS_5	LDO output voltage 1V mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select LDO output voltage 1V */
```

```
pmu_ldo_output_select(PMU_LDOVS_5);
```

### pmu\_slido\_output\_select

The description of pmu\_slido\_output\_select is shown as below:

Table 3-854. Function pmu\_slido\_output\_select

Function name	pmu_slido_output_select
Function prototype	void pmu_slido_output_select(uint32_t slido_n);
Function descriptions	Deep-sleep mode V0.9V core voltage select
Precondition	-
The called functions	-
Input parameter{in}	
slido_n	select Stop mode voltage
PMU_SLDOVS_0	SLDOVS scale 0.6V
PMU_SLDOVS_1	SLDOVS scale 0.7V
PMU_SLDOVS_2	SLDOVS scale 0.8V
PMU_SLDOVS_3	SLDOVS scale 0.9V
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* select Deep-sleep mode voltage 0.9V */
pmu_sldo_output_select(PMU_SLDOVS_3);
```

### pmu\_vbat\_charging\_select

The description of pmu\_vbat\_charging\_select is shown as below:

**Table 3-855. Function pmu\_vbat\_charging\_select**

<b>Function name</b>	pmu_vbat_charging_select
<b>Function prototype</b>	void pmu_vbat_charging_select(uint32_t resistor);
<b>Function descriptions</b>	PMU V <sub>BAT</sub> battery charging resistor selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>resistor</b>	select PMU VBAT battery charging resistor
<i>PMU_VCRSEL_5K</i>	5 kOhms resistor is selected for charging VBAT battery
<i>PMU_VCRSEL_1P5K</i>	1.5 kOhms resistor is selected for charging VBAT battery
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select PMU VBAT battery charging resistor to 1.5 kOhms */
pmu_vbat_charging_select(PMU_VCRSEL_1P5K);
```

### pmu\_vbat\_charging\_enable

The description of pmu\_vbat\_charging\_enable is shown as below:

**Table 3-856. Function pmu\_vbat\_charging\_enable**

<b>Function name</b>	pmu_vbat_charging_enable
<b>Function prototype</b>	void pmu_vbat_charging_enable(void);
<b>Function descriptions</b>	enable V <sub>BAT</sub> battery charging
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable VBAT battery charging */
pmu_vbat_charging_enable();
```

### pmu\_vbat\_charging\_disable

The description of pmu\_vbat\_charging\_disable is shown as below:

**Table 3-857. Function pmu\_vbat\_charging\_disable**

<b>Function name</b>	pmu_vbat_charging_disable
<b>Function prototype</b>	void pmu_vbat_charging_disable(void);
<b>Function descriptions</b>	disable V <sub>BAT</sub> battery charging
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable VBAT battery charging */
pmu_vbat_charging_disable();
```

### pmu\_vbat\_temp\_monitor\_enable

The description of pmu\_vbat\_temp\_monitor\_enable is shown as below:

**Table 3-858. Function pmu\_vbat\_temp\_monitor\_enable**

<b>Function name</b>	pmu_vbat_temp_monitor_enable
<b>Function prototype</b>	void pmu_vbat_temp_monitor_enable(void);
<b>Function descriptions</b>	enable V <sub>BAT</sub> and temperature monitoring
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable VBAT and temperature monitoring */
```

```
pmu_vbat_temp_monitor_enable();
```

### pmu\_vbat\_temp\_monitor\_disable

The description of pmu\_vbat\_temp\_monitor\_disable is shown as below:

**Table 3-859. Function pmu\_vbat\_temp\_monitor\_disable**

<b>Function name</b>	pmu_vbat_temp_monitor_disable
<b>Function prototype</b>	void pmu_vbat_temp_monitor_disable(void);
<b>Function descriptions</b>	disable VBAT and temperature monitoring
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable VBAT and temperature monitoring */
```

```
pmu_vbat_temp_monitor_disable();
```

### pmu\_usb\_regulator\_enable

The description of pmu\_usb\_regulator\_enable is shown as below:

**Table 3-860. Function pmu\_usb\_regulator\_enable**

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

Example:

```
/* enable USB regulator */
```

```
pmu_usb_regulator_enable();
```

## pmu\_usb\_regulator\_disable

The description of pmu\_usb\_regulator\_disable is shown as below:

**Table 3-861. Function pmu\_usb\_regulator\_disable**

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

Example:

```
/* disable USB regulator */
pmu_usb_regulator_disable();
```

## pmu\_usb\_voltage\_detector\_enable

The description of pmu\_usb\_voltage\_detector\_enable is shown as below:

**Table 3-862. Function pmu\_usb\_voltage\_detector\_enable**

<b>Function name</b>	pmu_usb_voltage_detector_enable
<b>Function prototype</b>	void pmu_usb_voltage_detector_enable(void);
<b>Function descriptions</b>	enable V <sub>DD33USB</sub> voltage level detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable VDD33USB voltage level detector */
pmu_usb_voltage_detector_enable();
```

## pmu\_usb\_voltage\_detector\_disable

The description of pmu\_usb\_voltage\_detector\_disable is shown as below:

**Table 3-863. Function pmu\_usb\_voltage\_detector\_disable**

<b>Function name</b>	pmu_usb_voltage_detector_disable
<b>Function prototype</b>	void pmu_usb_voltage_detector_disable(void);
<b>Function descriptions</b>	disable V <sub>DD33USB</sub> voltage level detector
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable VDD33USB voltage level detector */
```

```
pmu_usb_voltage_detector_disable();
```

### pmu\_smpps\_ldo\_supply\_config

The description of pmu\_smpps\_ldo\_supply\_config is shown as below:

**Table 3-864. Function pmu\_smpps\_ldo\_supply\_config**

<b>Function name</b>	pmu_smpps_ldo_supply_config
<b>Function prototype</b>	void pmu_smpps_ldo_supply_config(uint32_t smpsmode);
<b>Function descriptions</b>	power supply configurations
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>smpsmode</b>	power supply mode
<i>PMU_LDO_SUPPLY</i>	V <sub>0.9v</sub> domains are supplied from the LDO
<i>PMU_DIRECT_SMPS_SUPPLY</i>	V <sub>0.9v</sub> domains are supplied from the SMPS only
<i>PMU_BYPASS</i>	The SMPS disabled and the LDO Bypass. The V <sub>0.9v</sub> domains are supplied from an external source
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure V0.9v domain Bypass */
```

```
pmu_smpps_ldo_supply_config(PMU_BYPASS);
```

## pmu\_to\_sleepmode

The description of pmu\_to\_sleepmode is shown as below:

**Table 3-865. Function pmu\_to\_sleepmode**

<b>Function name</b>	pmu_to_sleepmode
<b>Function prototype</b>	void pmu_to_sleepmode(uint8_t sleepmodecmd);
<b>Function descriptions</b>	enter sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>sleepmodecmd</b>	command to enter sleep mode
<i>WFI_CMD</i>	use WFI command
<i>WFE_CMD</i>	use WFE command
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* PMU work at sleep mode */
pmu_to_sleepmode(WFI_CMD);
```

## pmu\_to\_deepsleepmode

The description of pmu\_to\_deepsleepmode is shown as below:

**Table 3-866. Function pmu\_to\_deepsleepmode**

<b>Function name</b>	pmu_to_deepsleepmode
<b>Function prototype</b>	void pmu_to_deepsleepmode(uint8_t deepsleepmodecmd);
<b>Function descriptions</b>	enter deepsleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>deepsleepmodecmd</b>	command to enter deepsleep mode
<i>WFI_CMD</i>	use WFI command
<i>WFE_CMD</i>	use WFE command
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* PMU work in deepsleep mode */
```

```
pmu_to_deepsleepmode(WFI_CMD);
```

### pmu\_to\_standbymode

The description of pmu\_to\_standbymode is shown as below:

**Table 3-867. Function pmu\_to\_standbymode**

<b>Function name</b>	pmu_to_standbymode
<b>Function prototype</b>	void pmu_to_standbymode(void);
<b>Function descriptions</b>	enter standby mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* PMU work at standby mode */
```

```
pmu_to_standby();
```

### pmu\_wakeup\_pin\_enable

The description of pmu\_wakeup\_pin\_enable is shown as below:

**Table 3-868. Function pmu\_wakeup\_pin\_enable**

<b>Function name</b>	pmu_wakeup_pin_enable
<b>Function prototype</b>	void pmu_wakeup_pin_enable(uint32_t wakeup_pin);
<b>Function descriptions</b>	enable PMU wakeup pin
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wakeup_pin</b>	Wakeup pin
<i>PMU_WAKEUP_PIN0</i>	WKUP Pin 0 (PA0)
<i>PMU_WAKEUP_PIN1</i>	WKUP Pin 1 (PA2)
<i>PMU_WAKEUP_PIN3</i>	WKUP Pin 3 (PC13)
<i>PMU_WAKEUP_PIN5</i>	WKUP Pin 5 (PC1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-



Example:

```
/* enable wakeup pin0 */
pmu_wakeup_pin_enable(PMU_WAKEUP_PIN0);
```

### pmu\_wakeup\_pin\_disable

The description of pmu\_wakeup\_pin\_disable is shown as below:

**Table 3-869. Function pmu\_wakeup\_pin\_disable**

<b>Function name</b>	pmu_wakeup_pin_disable
<b>Function prototype</b>	void pmu_wakeup_pin_disable(uint32_t wakeup_pin);
<b>Function descriptions</b>	disable PMU wakeup pin
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wakeup_pin</b>	Wakeup pin
PMU_WAKEUP_PIN0	WKUP Pin 0 (PA0)
PMU_WAKEUP_PIN1	WKUP Pin 1 (PA2)
PMU_WAKEUP_PIN3	WKUP Pin 3 (PC13)
PMU_WAKEUP_PIN5	WKUP Pin 5 (PC1)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable wakeup pin0 */
pmu_wakeup_pin_disable(PMU_WAKEUP_PIN0);
```

### pmu\_backup\_write\_enable

The description of pmu\_backup\_write\_enable is shown as below:

**Table 3-870. Function pmu\_backup\_write\_enable**

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

-	-
---	---

Example:

```
/* enable backup domain write */
pmu_backup_write_enable();
```

### pmu\_backup\_write\_disable

The description of pmu\_backup\_write\_disable is shown as below:

**Table 3-871. Function pmu\_backup\_write\_disable**

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

Example:

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

### pmu\_backup\_voltage\_stabilizer\_enable

The description of pmu\_backup\_voltage\_stabilizer\_enable is shown as below:

**Table 3-872. Function pmu\_backup\_voltage\_stabilizer\_enable**

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

Example:

```
/* enable backup voltage stabilizer */
```

```
pmu_backup_voltage_stabilizer_enable();
```

### pmu\_backup\_voltage\_stabilizer\_disable

The description of pmu\_backup\_voltage\_stabilizer\_disable is shown as below:

**Table 3-873. Function pmu\_backup\_voltage\_stabilizer\_disable**

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

Example:

```
/* disable backup voltage stabilizer */
```

```
pmu_backup_voltage_stabilizer_disable();
```

### pmu\_enter\_deepsleep\_wait\_time\_config

The description of pmu\_enter\_deepsleep\_wait\_time\_config is shown as below:

**Table 3-874. Function pmu\_enter\_deepsleep\_wait\_time\_config**

<b>Function name</b>	pmu_enter_deepsleep_wait_time_config
<b>Function prototype</b>	void pmu_enter_deepsleep_wait_time_config (int32_t wait_time);
<b>Function descriptions</b>	configure IRC counter before enter Deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wait_time</b>	IRC counter before enter Deep-sleep mode(0x00~0x1F)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure IRC counter before enter Deep-sleep mode to 0x10 */
```

```
pmu_enter_deepsleep_wait_time_config(0x10);
```

## pmu\_exit\_deepsleep\_wait\_time\_config

The description of pmu\_exit\_deepsleep\_wait\_time\_config is shown as below:

**Table 3-875. Function pmu\_exit\_deepsleep\_wait\_time\_config**

<b>Function name</b>	pmu_exit_deepsleep_wait_time_config
<b>Function prototype</b>	void pmu_exit_deepsleep_wait_time_config (int32_t wait_time);
<b>Function descriptions</b>	IRC counter before exit Deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wait_time</b>	IRC counter before exit Deep-sleep mode(0x00~0xFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure IRC counter before exit Deep-sleep mode to 0x10 */
```

```
pmu_exit_deepsleep_wait_time_config(0x10);
```

## pmu\_flag\_get

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

**Table 3-876. 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_LVDF</i>	low voltage detector status flag
<i>PMU_FLAG_VAVDF</i>	V <sub>DDA</sub> analog voltage detector voltage output on V <sub>DDA</sub> flag
<i>PMU_FLAG_VOVD</i>	peripheral voltage on V <sub>DDA</sub> detector flag
<i>PMU_FLAG_VBATLF</i>	V <sub>BAT</sub> level monitoring versus low threshold
<i>PMU_FLAG_VBATHF</i>	V <sub>BAT</sub> level monitoring versus high threshold
<i>PMU_FLAG_TEMPLF</i>	temperature level monitoring versus low threshold
<i>PMU_FLAG_TEMP</i>	temperature level monitoring versus high threshold
<i>PMU_FLAG_DVSRF</i>	step-down voltage stabilizer ready flag bit
<i>PMU_FLAG_USB33RF</i>	USB supply ready flag bit

<i>PMU_FLAG_PWRRF</i>	power Ready flag bit.
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

```
FlagStatus status;
```

```
status = pmu_flag_get(PMU_FLAG_WAKEUP);
```

### pmu\_flag\_clear

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

**Table 3-877. 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>	flag
<i>PMU_FLAG_WAKEUP</i>	wakeup flag
<i>PMU_FLAG_STANDBY</i>	standby flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear flag bit */
```

```
pmu_flag_clear(PMU_FLAG_STANDBY);
```

## 3.27. RAMECCMU

RAMECCMU is the RAM ECC monitor units which provide a mean for application software to verify ECC status and execute service routines when an error occurs. The RAMECCMU registers are listed in chapter [3.27.1](#), the RAMECCMU firmware functions are introduced in chapter [3.27.2](#).

### 3.27.1. Descriptions of Peripheral registers

**Table 3-878. RAMECCMU Registers**

Registers	Descriptions
RAMECCMU_INT	RAMECCMU global interrupt register
RAMECCMU_MxCTL	RAMECCMU monitor x control register
RAMECCMU_MxSTAT	RAMECCMU monitor x status register
RAMECCMU_MxFAADDR	RAMECCMU monitor x failing address register
RAMECCMU_MxFDL	RAMECCMU monitor x failing data low register
RAMECCMU_MxFDH	RAMECCMU monitor x failing data high register
RAMECCMU_MxFECODE	RAMECCMU monitor x failing ECC error code register

### 3.27.2. Descriptions of Peripheral functions

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

**Table 3-879. RAMECCMU firmware function**

Function name	Function description
rameccmu_deinit	deinit RAMECCMU unit
rameccmu_monitor_failing_address_get	get RAMECCMU monitor ECC failing address
rameccmu_monitor_failing_data_low_bits_get	get RAMECCMU monitor ECC failing data low 32 bits
rameccmu_monitor_failing_data_high_bits_get	get RAMECCMU monitor ECC failing data high 32 bits
rameccmu_monitor_failing_ecc_error_code_get	get RAMECCMU monitor failing ECC error code
rameccmu_global_interrupt_enable	enable RAMECCMU global ECC interrupt
rameccmu_global_interrupt_disable	disable RAMECCMU global ECC interrupt
rameccmu_monitor_interrupt_enable	enable RAMECCMU monitor ECC error interrupt
rameccmu_monitor_interrupt_disable	disable RAMECCMU monitor ECC error interrupt
rameccmu_monitor_flag_get	get RAMECCMU monitor ECC error flag
rameccmu_monitor_flag_clear	clear RAMECCMU monitor ECC error flag
rameccmu_monitor_interrupt_flag_get	get RAMECCMU monitor ECC interrupt error flag
rameccmu_monitor_interrupt_flag_clear	clear RAMECCMU monitor interrupt ECC error flag

## Enum rameccmu\_monitor\_enum

**Table 3-880. Enum rameccmu\_monitor\_enum**

enum name	Function description
RAMECCMU0_MONIT OR0	RAMECCMU0 monitor 0
RAMECCMU0_MONIT OR1	RAMECCMU0 monitor 1
RAMECCMU0_MONIT OR2	RAMECCMU0 monitor 2
RAMECCMU0_MONIT OR3	RAMECCMU0 monitor 3
RAMECCMU0_MONIT OR4	RAMECCMU0 monitor 4
RAMECCMU1_MONIT OR0	RAMECCMU1 monitor 0
RAMECCMU1_MONIT OR1	RAMECCMU1 monitor 1
RAMECCMU1_MONIT OR2	RAMECCMU1 monitor 2

## rameccmu\_deinit

The description of rameccmu\_deinit is shown as below:

**Table 3-881. Function rameccmu\_deinit**

<b>Function name</b>	rameccmu_deinit
<b>Function prototype</b>	void rameccmu_deinit(uint32_t rameccmu_periph);
<b>Function descriptions</b>	deinit RAMECCMU unit
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_periph</b>	rameccmu
<i>RAMECCMU0</i>	RAMECCMU for Region 0
<i>RAMECCMU1</i>	RAMECCMU for Region 1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinit RAMECCMU0 unit */
rameccmu_deinit(RAMECCMU0);
```

## rameccmu\_monitor\_failing\_address\_get

The description of rameccmu\_monitor\_failing\_address\_get is shown as below:

**Table 3-882. Function rameccmu\_monitor\_failing\_address\_get**

<b>Function name</b>	rameccmu_monitor_failing_address_get
<b>Function prototype</b>	uint32_t rameccmu_monitor_failing_address_get(rameccmu_monitor_enum rameccmu_monitor);
<b>Function descriptions</b>	get RAMECCMU monitor ECC failing address
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_monitor</b>	rameccmu monitor
RAMECCMU0_MONIT OR0	RAMECCMU0 monitor 0
RAMECCMU0_MONIT OR1	RAMECCMU0 monitor 1
RAMECCMU0_MONIT OR2	RAMECCMU0 monitor 2
RAMECCMU0_MONIT OR3	RAMECCMU0 monitor 3
RAMECCMU0_MONIT OR4	RAMECCMU0 monitor 4
RAMECCMU1_MONIT OR0	RAMECCMU1 monitor 0
RAMECCMU1_MONIT OR1	RAMECCMU1 monitor 1
RAMECCMU1_MONIT OR2	RAMECCMU1 monitor 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	ECC error failing address

Example:

```
/* get RAMECCMU monitor ECC failing address */
```

```
uint32_t val;
```

```
val = rameccmu_monitor_failing_address_get(RAMECCMU0_MONITOR0);
```

## rameccmu\_monitor\_failing\_data\_low\_bits\_get

The description of rameccmu\_monitor\_failing\_data\_low\_bits\_get is shown as below:



**Table 3-883. Function `rameccmu_monitor_failing_data_low_bits_get`**

<b>Function name</b>	<code>rameccmu_monitor_failing_data_low_bits_get</code>
<b>Function prototype</b>	<pre>uint32_t rameccmu_monitor_failing_data_low_bits_get(rameccmu_monitor_enum rameccmu_monitor);</pre>
<b>Function descriptions</b>	get RAMECCMU monitor ECC failing data low 32 bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_monitor</b>	rameccmu monitor
<code>RAMECCMU0_MONIT OR0</code>	RAMECCMU0 monitor 0
<code>RAMECCMU0_MONIT OR1</code>	RAMECCMU0 monitor 1
<code>RAMECCMU0_MONIT OR2</code>	RAMECCMU0 monitor 2
<code>RAMECCMU0_MONIT OR3</code>	RAMECCMU0 monitor 3
<code>RAMECCMU0_MONIT OR4</code>	RAMECCMU0 monitor 4
<code>RAMECCMU1_MONIT OR0</code>	RAMECCMU1 monitor 0
<code>RAMECCMU1_MONIT OR1</code>	RAMECCMU1 monitor 1
<code>RAMECCMU1_MONIT OR2</code>	RAMECCMU1 monitor 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	ECC failing data low 32 bits

Example:

```
/* get RAMECCMU monitor ECC failing data low 32 bits */
```

```
uint32_t val_low;
```

```
val_low = rameccmu_monitor_failing_data_low_bits_get(RAMECCMU0_MONITOR0);
```

### **`rameccmu_monitor_failing_data_high_bits_get`**

The description of `rameccmu_monitor_failing_data_high_bits_get` is shown as below:

**Table 3-884. Function `rameccmu_monitor_failing_data_high_bits_get`**

<b>Function name</b>	<code>rameccmu_monitor_failing_data_high_bits_get</code>
<b>Function prototype</b>	<pre>uint32_t</pre>

	rameccmu_monitor_failing_data_high_bits_get(rameccmu_monitor_enum rameccmu_monitor);
<b>Function descriptions</b>	get RAMECCMU monitor ECC failing data high 32 bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_monitor</b>	rameccmu monitor
<i>RAMECCMU0_MONIT OR0</i>	RAMECCMU0 monitor 0
<i>RAMECCMU0_MONIT OR1</i>	RAMECCMU0 monitor 1
<i>RAMECCMU0_MONIT OR2</i>	RAMECCMU0 monitor 2
<i>RAMECCMU0_MONIT OR3</i>	RAMECCMU0 monitor 3
<i>RAMECCMU0_MONIT OR4</i>	RAMECCMU0 monitor 4
<i>RAMECCMU1_MONIT OR0</i>	RAMECCMU1 monitor 0
<i>RAMECCMU1_MONIT OR1</i>	RAMECCMU1 monitor 1
<i>RAMECCMU1_MONIT OR2</i>	RAMECCMU1 monitor 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	ECC failing data high 32 bits

Example:

```
/* get RAMECCMU monitor ECC failing data high 32 bits */
```

```
uint32_t val_high;
```

```
val_high = rameccmu_monitor_failing_data_high_bits_get(RAMECCMU0_MONITOR0);
```

### rameccmu\_monitor\_failing\_ecc\_error\_code\_get

The description of rameccmu\_monitor\_failing\_ecc\_error\_code\_get is shown as below:

**Table 3-885. Function rameccmu\_monitor\_failing\_ecc\_error\_code\_get**

<b>Function name</b>	rameccmu_monitor_failing_ecc_error_code_get
<b>Function prototype</b>	uint32_t rameccmu_monitor_failing_ecc_error_code_get(rameccmu_monitor_enum rameccmu_monitor);
<b>Function descriptions</b>	

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_monitor</b>	rameccmu monitor
<i>RAMECCMU0_MONIT</i> <i>OR0</i>	RAMECCMU0 monitor 0
<i>RAMECCMU0_MONIT</i> <i>OR1</i>	RAMECCMU0 monitor 1
<i>RAMECCMU0_MONIT</i> <i>OR2</i>	RAMECCMU0 monitor 2
<i>RAMECCMU0_MONIT</i> <i>OR3</i>	RAMECCMU0 monitor 3
<i>RAMECCMU0_MONIT</i> <i>OR4</i>	RAMECCMU0 monitor 4
<i>RAMECCMU1_MONIT</i> <i>OR0</i>	RAMECCMU1 monitor 0
<i>RAMECCMU1_MONIT</i> <i>OR1</i>	RAMECCMU1 monitor 1
<i>RAMECCMU1_MONIT</i> <i>OR2</i>	RAMECCMU1 monitor 2
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	ECC failing error code

Example:

```
/* get RAMECCMU monitor failing ECC error code */
```

```
uint32_t val;
```

```
val = rameccmu_monitor_failing_ecc_error_code_get(RAMECCMU0_MONITOR0);
```

### rameccmu\_global\_interrupt\_enable

The description of rameccmu\_global\_interrupt\_enable is shown as below:

**Table 3-886. Function rameccmu\_global\_interrupt\_enable**

<b>Function name</b>	rameccmu_global_interrupt_enable
<b>Function prototype</b>	void rameccmu_global_interrupt_enable(uint32_t rameccmu_periph, uint32_t interrupt);
<b>Function descriptions</b>	enable RAMECCMU global ECC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_periph</b>	rameccmu

<i>RAMECCMU0</i>	RAMECCMU for Region 0
<i>RAMECCMU1</i>	RAMECCMU for Region 1
<b>Input parameter{in}</b>	
<b>interrupt</b>	global ECC interrupt
<i>RAMECCMU_INT_EC_C_GLOBAL_ERROR</i>	ECC global error interrupt
<i>RAMECCMU_INT_EC_C_SINGLE_ERROR</i>	ECC single error interrupt
<i>RAMECCMU_INT_EC_C_DOUBLE_ERROR</i>	ECC double error interrupt
<i>RAMECCMU_INT_EC_C_DOUBLE_ERROR_BYTE_WRITE</i>	ECC double error on byte write interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable ECC global error interrupt */
```

```
rameccmu_global_interrupt_enable(RAMECCMU0,  
RAMECCMU_INT_ECC_GLOBAL_ERROR);
```

### rameccmu\_global\_interrupt\_disable

The description of `rameccmu_global_interrupt_disable` is shown as below:

**Table 3-887. Function `rameccmu_global_interrupt_disable`**

<b>Function name</b>	<code>rameccmu_global_interrupt_disable</code>
<b>Function prototype</b>	<code>void rameccmu_global_interrupt_disable(uint32_t rameccmu_periph, uint32_t interrupt);</code>
<b>Function descriptions</b>	disable RAMECCMU global ECC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_periph</b>	rameccmu
<i>RAMECCMU0</i>	RAMECCMU for Region 0
<i>RAMECCMU1</i>	RAMECCMU for Region 1
<b>Input parameter{in}</b>	
<b>interrupt</b>	global ECC interrupt
<i>RAMECCMU_INT_EC_C_GLOBAL_ERROR</i>	ECC global error interrupt
<i>RAMECCMU_INT_EC_C_SINGLE_ERROR</i>	ECC single error interrupt

<i>RAMECCMU_INT_EC</i> <i>C_DOUBLE_ERROR</i>	ECC double error interrupt
<i>RAMECCMU_INT_EC</i> <i>C_DOUBLE_ERROR_</i> <i>BYTE_WRITE</i>	ECC double error on byte write interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable ECC global error interrupt */
```

```
rameccmu_global_interrupt_disable(RAMECCMU0,  
RAMECCMU_INT_ECC_GLOBAL_ERROR);
```

### rameccmu\_monitor\_interrupt\_enable

The description of `rameccmu_monitor_interrupt_enable` is shown as below:

**Table 3-888. Function `rameccmu_monitor_interrupt_enable`**

<b>Function name</b>	<code>rameccmu_monitor_interrupt_enable</code>
<b>Function prototype</b>	<code>void rameccmu_monitor_interrupt_enable(rameccmu_monitor_enum rameccmu_monitor, uint32_t monitor_interrupt);</code>
<b>Function descriptions</b>	enable RAMECCMU monitor ECC error interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_monitor</b>	rameccmu monitor
<i>RAMECCMU0_MONIT</i> <i>OR0</i>	RAMECCMU0 monitor 0
<i>RAMECCMU0_MONIT</i> <i>OR1</i>	RAMECCMU0 monitor 1
<i>RAMECCMU0_MONIT</i> <i>OR2</i>	RAMECCMU0 monitor 2
<i>RAMECCMU0_MONIT</i> <i>OR3</i>	RAMECCMU0 monitor 3
<i>RAMECCMU0_MONIT</i> <i>OR4</i>	RAMECCMU0 monitor 4
<i>RAMECCMU1_MONIT</i> <i>OR0</i>	RAMECCMU1 monitor 0
<i>RAMECCMU1_MONIT</i> <i>OR1</i>	RAMECCMU1 monitor 1
<i>RAMECCMU1_MONIT</i> <i>OR2</i>	RAMECCMU1 monitor 2

Input parameter{in}	
<b>monitor_interrupt</b>	monitor interrupt
<i>RAMECCMU_INT_EC_C_SINGLE_ERROR</i>	ECC single error interrupt
<i>RAMECCMU_INT_EC_C_DOUBLE_ERROR</i>	ECC double error interrupt
<i>RAMECCMU_INT_EC_C_DOUBLE_ERROR_BYTE_WRITE</i>	ECC double error on byte write interrupt
<i>RAMECCMU_INT_EC_C_ERROR_LATCHING</i>	ECC error latching
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable RAMECCMU monitor ECC error interrupt */
```

```
rameccmu_monitor_interrupt_enable(RAMECCMU0_MONITOR0,  
RAMECCMU_INT_ECC_SINGLE_ERROR);
```

### rameccmu\_monitor\_interrupt\_disable

The description of rameccmu\_monitor\_interrupt\_disable is shown as below:

**Table 3-889. Function rameccmu\_monitor\_interrupt\_disable**

Function name	rameccmu_monitor_interrupt_disable
Function prototype	void rameccmu_monitor_interrupt_disable(rameccmu_monitor_enum rameccmu_monitor, uint32_t monitor_interrupt);
Function descriptions	disable RAMECCMU monitor ECC error interrupt
Precondition	-
The called functions	-
Input parameter{in}	
<b>rameccmu_monitor</b>	rameccmu monitor
<i>RAMECCMU0_MONITOR0</i>	RAMECCMU0 monitor 0
<i>RAMECCMU0_MONITOR1</i>	RAMECCMU0 monitor 1
<i>RAMECCMU0_MONITOR2</i>	RAMECCMU0 monitor 2
<i>RAMECCMU0_MONITOR3</i>	RAMECCMU0 monitor 3
<i>RAMECCMU0_MONITOR4</i>	RAMECCMU0 monitor 4

<i>RAMECCMU1_MONIT</i> <i>OR0</i>	RAMECCMU1 monitor 0
<i>RAMECCMU1_MONIT</i> <i>OR1</i>	RAMECCMU1 monitor 1
<i>RAMECCMU1_MONIT</i> <i>OR2</i>	RAMECCMU1 monitor 2
<b>Input parameter{in}</b>	
<b>monitor_interrupt</b>	monitor interrupt
<i>RAMECCMU_INT_EC</i> <i>C_SINGLE_ERROR</i>	ECC single error interrupt
<i>RAMECCMU_INT_EC</i> <i>C_DOUBLE_ERROR</i>	ECC double error interrupt
<i>RAMECCMU_INT_EC</i> <i>C_DOUBLE_ERROR_</i> <i>BYTE_WRITE</i>	ECC double error on byte write interrupt
<i>RAMECCMU_INT_EC</i> <i>C_ERROR_LATCHING</i>	ECC error latching
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable RAMECCMU monitor ECC error interrupt */
```

```
rameccmu_monitor_interrupt_disable(RAMECCMU0_MONITOR0,  
RAMECCMU_INT_ECC_SINGLE_ERROR);
```

### rameccmu\_monitor\_flag\_get

The description of rameccmu\_monitor\_flag\_get is shown as below:

**Table 3-890. Function rameccmu\_monitor\_flag\_get**

<b>Function name</b>	rameccmu_monitor_flag_get
<b>Function prototype</b>	FlagStatus rameccmu_monitor_flag_get(rameccmu_monitor_enum rameccmu_monitor, uint32_t flag);
<b>Function descriptions</b>	get RAMECCMU monitor ECC error flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_monitor</b>	rameccmu monitor
<i>RAMECCMU0_MONIT</i> <i>OR0</i>	RAMECCMU0 monitor 0
<i>RAMECCMU0_MONIT</i> <i>OR1</i>	RAMECCMU0 monitor 1

RAMECCMU0_MONIT OR2	RAMECCMU0 monitor 2
RAMECCMU0_MONIT OR3	RAMECCMU0 monitor 3
RAMECCMU0_MONIT OR4	RAMECCMU0 monitor 4
RAMECCMU1_MONIT OR0	RAMECCMU1 monitor 0
RAMECCMU1_MONIT OR1	RAMECCMU1 monitor 1
RAMECCMU1_MONIT OR2	RAMECCMU1 monitor 2
<b>Input parameter{in}</b>	
<b>flag</b>	RAMECCMU monitor flag
RAMECCMU_FLAG_E CC_SINGLE_ERROR	ECC single error detected and corrected flag
RAMECCMU_FLAG_E CC_DOUBLE_ERROR	ECC double error detected flag
RAMECCMU_FLAG_E CC_DOUBLE_ERROR _BYTE_WRITE	ECC double error on byte write detected flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	RESET or SET

Example:

```
/* get RAMECCMU monitor ECC error flag */
```

```
FlagStatus flag;
```

```
flag = rameccmu_monitor_flag_get(RAMECCMU0_MONITOR0,  
RAMECCMU_FLAG_ECC_SINGLE_ERROR);
```

### rameccmu\_monitor\_flag\_clear

The description of rameccmu\_monitor\_flag\_clear is shown as below:

**Table 3-891. Function rameccmu\_monitor\_flag\_clear**

<b>Function name</b>	rameccmu_monitor_flag_clear
<b>Function prototype</b>	void rameccmu_monitor_flag_clear(rameccmu_monitor_enum rameccmu_monitor, uint32_t flag);
<b>Function descriptions</b>	clear RAMECCMU monitor ECC error flag
<b>Precondition</b>	-
<b>The called functions</b>	-



Input parameter{in}	
<b>rameccmu_monitor</b>	rameccmu monitor
<i>RAMECCMU0_MONIT</i> <i>OR0</i>	RAMECCMU0 monitor 0
<i>RAMECCMU0_MONIT</i> <i>OR1</i>	RAMECCMU0 monitor 1
<i>RAMECCMU0_MONIT</i> <i>OR2</i>	RAMECCMU0 monitor 2
<i>RAMECCMU0_MONIT</i> <i>OR3</i>	RAMECCMU0 monitor 3
<i>RAMECCMU0_MONIT</i> <i>OR4</i>	RAMECCMU0 monitor 4
<i>RAMECCMU1_MONIT</i> <i>OR0</i>	RAMECCMU1 monitor 0
<i>RAMECCMU1_MONIT</i> <i>OR1</i>	RAMECCMU1 monitor 1
<i>RAMECCMU1_MONIT</i> <i>OR2</i>	RAMECCMU1 monitor 2
Input parameter{in}	
<b>flag</b>	RAMECCMU monitor flag
<i>RAMECCMU_FLAG_E</i> <i>CC_SINGLE_ERROR</i>	ECC single error detected and corrected flag
<i>RAMECCMU_FLAG_E</i> <i>CC_DOUBLE_ERROR</i>	ECC double error detected flag
<i>RAMECCMU_FLAG_E</i> <i>CC_DOUBLE_ERROR</i> <i>_BYTE_WRITE</i>	ECC double error on byte write detected flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear RAMECCMU monitor ECC error flag */
```

```
rameccmu_monitor_flag_clear(RAMECCMU0_MONITOR0,  
RAMECCMU_FLAG_ECC_SINGLE_ERROR);
```

### rameccmu\_monitor\_interrupt\_flag\_get

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

**Table 3-892. Function rameccmu\_monitor\_interrupt\_flag\_get**

<b>Function name</b>	rameccmu_monitor_interrupt_flag_get
<b>Function prototype</b>	FlagStatus

	<pre> rameccmu_monitor_interrupt_flag_get(rameccmu_monitor_enum rameccmu_monitor, uint32_t flag); </pre>
<b>Function descriptions</b>	get RAMECCMU monitor ECC interrupt error flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_monitor</b>	rameccmu monitor
<i>RAMECCMU0_MONIT OR0</i>	RAMECCMU0 monitor 0
<i>RAMECCMU0_MONIT OR1</i>	RAMECCMU0 monitor 1
<i>RAMECCMU0_MONIT OR2</i>	RAMECCMU0 monitor 2
<i>RAMECCMU0_MONIT OR3</i>	RAMECCMU0 monitor 3
<i>RAMECCMU0_MONIT OR4</i>	RAMECCMU0 monitor 4
<i>RAMECCMU1_MONIT OR0</i>	RAMECCMU1 monitor 0
<i>RAMECCMU1_MONIT OR1</i>	RAMECCMU1 monitor 1
<i>RAMECCMU1_MONIT OR2</i>	RAMECCMU1 monitor 2
<b>Input parameter{in}</b>	
<b>flag</b>	RAMECCMU monitor flag
<i>RAMECCMU_INT_FL G_ECC_SINGLE_ERR OR</i>	ECC single error detected and corrected flag
<i>RAMECCMU_INT_FL G_ECC_DOUBLE_ER ROR</i>	ECC double error detected flag
<i>RAMECCMU_INT_FL G_ECC_DOUBLE_ER ROR_BYTE_WRITE</i>	ECC double error on byte write detected flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	RESET or SET

Example:

```
/* get RAMECCMU monitor ECC error flag */
```

```
FlagStatus flag;
```

```
flag = rameccmu_monitor_interrupt_flag_get(RAMECCMU0_MONITOR0,
```

RAMECCMU\_INT\_FLAG\_ECC\_SINGLE\_ERROR);

### rameccmu\_monitor\_interrupt\_flag\_clear

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

**Table 3-893. Function rameccmu\_monitor\_interrupt\_flag\_clear**

<b>Function name</b>	rameccmu_monitor_interrupt_flag_clear
<b>Function prototype</b>	void rameccmu_monitor_interrupt_flag_clear(rameccmu_monitor_enum rameccmu_monitor, uint32_t flag);
<b>Function descriptions</b>	clear RAMECCMU monitor interrupt ECC error flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rameccmu_monitor</b>	rameccmu monitor
RAMECCMU0_MONIT OR0	RAMECCMU0 monitor 0
RAMECCMU0_MONIT OR1	RAMECCMU0 monitor 1
RAMECCMU0_MONIT OR2	RAMECCMU0 monitor 2
RAMECCMU0_MONIT OR3	RAMECCMU0 monitor 3
RAMECCMU0_MONIT OR4	RAMECCMU0 monitor 4
RAMECCMU1_MONIT OR0	RAMECCMU1 monitor 0
RAMECCMU1_MONIT OR1	RAMECCMU1 monitor 1
RAMECCMU1_MONIT OR2	RAMECCMU1 monitor 2
<b>Input parameter{in}</b>	
<b>flag</b>	RAMECCMU monitor flag
RAMECCMU_INT_FL G_ECC_SINGLE_ERR OR	ECC single error detected and corrected flag
RAMECCMU_INT_FL G_ECC_DOUBLE_ER ROR	ECC double error detected flag
RAMECCMU_INT_FL G_ECC_DOUBLE_ER ROR_BYTE_WRITE	ECC double error on byte write detected flag
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* clear RAMECCMU monitor ECC error flag */
```

```
rameccmu_monitor_interrupt_flag_clear(RAMECCMU0_MONITOR0,  
RAMECCMU_INT_FLAG_ECC_SINGLE_ERROR);
```

## 3.28. 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 [3.28.1](#), the RCU firmware functions are introduced in chapter [3.28.2](#).

### 3.28.1. Descriptions of Peripheral registers

RCU registers are listed in the table shown as below:

**Table 3-894. RCU Registers**

Registers	Descriptions
RCU_CTL	control register
RCU_PLL0	PLL0 register
RCU_CFG0	clock configuration register 0
RCU_INT	clock interrupt register
RCU_AHB1RST	AHB1 reset register
RCU_AHB2RST	AHB2 reset register
RCU_AHB3RST	AHB3 reset register
RCU_AHB4RST	AHB4 reset register
RCU_APB1RST	APB1 reset register
RCU_APB2RST	APB2 reset register
RCU_APB3RST	APB3 reset register
RCU_APB4RST	APB4 reset register
RCU_AHB1EN	AHB1 enable register
RCU_AHB2EN	AHB2 enable register
RCU_AHB3EN	AHB3 enable register
RCU_AHB4EN	AHB4 enable register
RCU_APB1EN	APB1 enable register
RCU_APB2EN	APB2 enable register
RCU_APB3EN	APB3 enable register
RCU_APB4EN	APB4 enable register
RCU_AHB1SPEN	AHB1 sleep mode enable register
RCU_AHB2SPEN	AHB2 sleep mode enable register

Registers	Descriptions
RCU_AHB3SPEN	AHB3 sleep mode enable register
RCU_AHB4SPEN	AHB4 sleep mode enable register
RCU_APB1SPEN	APB1 sleep mode enable register
RCU_APB2SPEN	APB2 sleep mode enable register
RCU_APB3SPEN	APB3 sleep mode enable register
RCU_APB4SPEN	APB4 sleep mode enable register
RCU_BDCTL	backup domain control register
RCU_RSTSCK	reset source / clock register
RCU_PLLADDCTL	PLL clock additional control register
RCU_PLL1	PLL1 register
RCU_PLL2	PLL2 register
RCU_CFG1	clock configuration register 1
RCU_CFG2	clock configuration register 2
RCU_CFG3	clock configuration register 3
RCU_PLLALL	PLL configuration register
RCU_PLL0FRA	PLL0 fraction configuration register
RCU_PLL1FRA	PLL1 fraction configuration register
RCU_PLL2FRA	PLL2 fraction configuration register
RCU_ADDCTL0	additional clock control register 0
RCU_ADDCTL1	additional clock control register 1
RCU_ADDINT	additional clock interrupt register
RCU_CFG4	clock configuration register 4
RCU_USBCLKCTL	USB clock control register
RCU_PLLUSBCFG	PLLUSB configuration register
RCU_ADDAPB2RS T	APB2 additional reset register
RCU_ADDAPB2EN	APB2 additional enable register
RCU_ADDAPB2SP EN	APB2 additional sleep mode enable register
RCU_CFG5	clock configuration register 5

### 3.28.2. Descriptions of Peripheral functions

**Table 3-895. RCU firmware function**

Function name	Function description
rcu_deinit	deinitialize the RCU
rcu_periph_clock_enable	enable the peripherals clock
rcu_periph_clock_disable	disable the peripherals clock
rcu_periph_clock_sleep_enable	enable the peripherals clock when sleep mode
rcu_periph_clock_sleep_disable	disable the peripherals clock when sleep mode
rcu_periph_reset_enable	reset the peripherals
rcu_periph_reset_disable	disable reset the peripheral

Function name	Function description
rcu_bkp_reset_enable	enable the BKP domain reset
rcu_bkp_reset_disable	disable the BKP domain reset
rcu_system_clock_source_config	configure the system clock source
rcu_system_clock_source_get	get the system clock source
rcu_ahb_clock_config	configure the AHB clock prescaler selection
rcu_apb1_clock_config	configure the APB1 clock prescaler selection
rcu_apb2_clock_config	configure the APB2 clock prescaler selection
rcu_apb3_clock_config	configure the APB3 clock prescaler selection
rcu_apb4_clock_config	configure the APB4 clock prescaler selection
rcu_ckout0_config	configure the CK_OUT0 clock source and divider
rcu_ckout1_config	configure the CK_OUT1 clock source and divider
rcu_pll_input_output_clock_range_config	configure the pll input and output clock range
rcu_pll_fractional_config	configure fractional part of the multiplication factor for PLL VCO
rcu_pll_fractional_latch_enable	enable PLL fractional latch
rcu_pll_fractional_latch_disable	disable PLL fractional latch
rcu_pll_source_config	configure the PLLs clock source selection
rcu_pll0_config	configure the PLL0
rcu_pll1_config	configure the PLL1 clock
rcu_pll2_config	configure the PLL2 clock
rcu_pll_clock_output_enable	enable the pllq pllq pllr divider output
rcu_pll_clock_output_disable	disable the pllq pllq pllr divider output
rcu_pllusb0_config	configure the PLLUSBHS0 clock
rcu_pllusb1_config	configure the PLLUSBHS1 clock
rcu_rtc_clock_config	configure the RTC clock source selection
rcu_rtc_div_config	configure the frequency division of RTC clock when HXTAL was selected as its clock source
rcu_ck48m_clock_config	configure the CK48M clock selection
rcu_pll48m_clock_config	configure the PLL48M clock selection
rcu_irc64mdiv_clock_config	configure the IRC64M clock divider selection
rcu_irc64mdiv_freq_get	get the irc64mdiv clock
rcu_timer_clock_prescaler_config	configure the TIMER clock prescaler selection
rcu_spi_clock_config	configure the SPI clock source selection
rcu_deepsleep_wakeup_sys_clock_config	configure the Deep-sleep wakeup system clock source selection
rcu_usart_clock_config	configure the USARTx(x=0,1,2,5) clock source selection
rcu_i2c_clock_config	configure the I2Cx(x=0,1,2,3) clock source selection
rcu_can_clock_config	configure the CANx(x=0,1,2) clock source selection
rcu_adc_clock_config	configure the ADCx(x=0,1,2) clock source selection
rcu_exmc_clock_config	configure the EXMC clock source selection
rcu_hpdf_clock_config	configure the HPDF clock source selection

Function name	Function description
rcu_per_clock_config	configure the peripheral clock source selection
rcu_usbhs_pll1qpsc_config	configure the PLL1Q prescaler
rcu_usb48m_clock_config	configure the USBHS48M clock source selection
rcu_usbhs_clock_config	configure the USBHS clock source selection
rcu_usbhs_clock_selection_enable	enable the USBHS clock source selection
rcu_usbhs_clock_selection_disable	disable configure the USBHS clock source selection
rcu_lxtal_drive_capability_config	configure the LXTAL drive capability
rcu_osci_stab_wait	wait for oscillator stabilization flags is SET or oscillator startup is timeout
rcu_osci_on	turn on the oscillator
rcu_osci_off	turn off the oscillator
rcu_osci_bypass_mode_enable	enable the oscillator bypass mode, HXTALEN or LXTALEN must be reset before it
rcu_osci_bypass_mode_disable	disable the oscillator bypass mode, HXTALEN or LXTALEN must be reset before it
rcu_irc64m_adjust_value_set	set the IRC64M adjust value
rcu_lpirc4m_adjust_value_set	set the LPIRC4M adjust value
rcu_hxtal_clock_monitor_enable	enable the HXTAL clock monitor
rcu_hxtal_clock_monitor_disable	disable the HXTAL clock monitor
rcu_lxtal_clock_monitor_enable	enable the LXTAL clock monitor
rcu_lxtal_clock_monitor_disable	disable the LXTAL clock monitor
rcu_clock_freq_get	get the system clock, bus and peripheral clock frequency
rcu_flag_get	get the clock stabilization and peripheral reset flags
rcu_all_reset_flag_clear	clear the reset flag
rcu_interrupt_enable	enable the stabilization interrupt
rcu_interrupt_disable	disable the stabilization interrupt
rcu_interrupt_flag_get	get the clock stabilization interrupt and ckm flags
rcu_interrupt_flag_clear	clear the interrupt flags

## Enum rcu\_periph\_enum

Table 3-896. Enum rcu\_periph\_enum

enum name	Function description
RCU_USBHS0	USBHS0 clock
RCU_USBHS0ULPI	USBHS0ULPI clock
RCU_DMA0	DMA0 clock
RCU_DMA1	DMA1 clock
RCU_DMAMUX	IPA clock
RCU_ENET0	ENET0 clock
RCU_USBHS1	USBHS1 clock
RCU_USBHS1ULPI	USBHS1HSULPI clock
RCU_FAC	FAC clock

enum name	Function description
RCU_TRNG	TRNG clock
RCU_TMU	TMU clock
RCU_RAMECCMU1	RAMECCMU1 clock
RCU_EXMC	EXMC clock
RCU_SDIO0	SDIO0 clock
RCU_MDMA	MDMMA clock
RCU_OSPIM	OSPIM clock
RCU_OSPI0	OSPI0 clock
RCU_OSPI1	OSPI1 clock
RCU_RTDEC0	RTDEC0 clock
RCU_RTDEC1	RTDEC1 clock
RCU_RAMECCMU0	RAMECCMU0 clock
RCU_CPU	CPU 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_GPIOH	GPIOH clock
RCU_GPIOJ	GPIOJ clock
RCU_GPIOK	GPIOK clock
RCU_BKPSRAM	BKPSRAM clock
RCU_CRC	CRC clock
RCU_TIMER1	TIMER1 clock
RCU_TIMER2	TIMER2 clock
RCU_TIMER3	TIMER3 clock
RCU_TIMER4	TIMER4 clock
RCU_TIMER5	TIMER5 clock
RCU_TIMER6	TIMER6 clock
RCU_TIMER22	TIMER22 clock
RCU_TIMER23	TIMER23 clock
RCU_TIMER50	TIMER50 clock
RCU_TIMER51	TIMER51 clock
RCU_SPI1	SPI1 clock
RCU_SPI2	SPI2 clock
RCU_USART1	USART1 clock
RCU_USART2	USART2 clock
RCU_UART3	UART3 clock
RCU_UART4	UART4 clock
RCU_I2C0	I2C0 clock



enum name	Function description
RCU_I2C1	I2C1 clock
RCU_I2C2	I2C2 clock
RCU_I2C3	I2C3 clock
RCU_CTC	CTC clock
RCU_DACHOLD	DACHOLD clock
RCU_DAC	DAC clock
RCU_UART6	UART6 clock
RCU_UART7	UART7 clock
RCU_TIMER0	TIMER0 clock
RCU_TIMER7	TIMER7 clock
RCU_USART0	USART0 clock
RCU_USART5	USART5 clock
RCU_ADC0	ADC0 clock
RCU_ADC1	ADC1 clock
RCU_ADC2	ADC2 clock
RCU_SPI0	SPI0 clock
RCU_SPI3	SPI3 clock
RCU_TIMER14	TIMER14 clock
RCU_TIMER15	TIMER15 clock
RCU_TIMER16	TIMER16 clock
RCU_HPDF	HPDF clock
RCU_SPI4	SPI4 clock
RCU_SPI5	SPI5 clock
RCU_TIMER40	TIMER40 clock
RCU_TIMER41	TIMER41 clock
RCU_TIMER42	TIMER42 clock
RCU_TIMER43	TIMER43 clock
RCU_TIMER44	TIMER44 clock
RCU_EDOUT	EDOUT clock
RCU_TRIGSEL	TRIGSEL clock
RCU_WWDGT	WWDGT clock
RCU_SYSCFG	SYSCFG clock
RCU_CMP	CMP clock
RCU_VREF	VREF clock
RCU_LPDTs	LPDTs clock
RCU_PMU	PMU clock
RCU_RTC	RTC clock
RCU_CAN0	CAN0 clock
RCU_CAN1	CAN1 clock
RCU_CAN2	CAN2 clock

## Enum rcu\_periph\_sleep\_enum

**Table 3-897. Enum rcu\_periph\_sleep\_enum**

enum name	Function description
RCU_USBHS0_SLP	USBHS0 clock
RCU_USBHS0ULPI_SLP	USBHS0ULPI clock
RCU_SRAM0_SLP	SRAM0 clock
RCU_SRAM1_SLP	SRAM1 clock
RCU_DMA0_SLP	DMA0 clock
RCU_DMA1_SLP	DMA1 clock
RCU_DMAMUX_SLP	DMAMUX clock
RCU_USBHS1_SLP	USBHS1 clock
RCU_USBHS1ULPI_SLP	USBHS1ULPI clock
RCU_FAC_SLP	FAC clock
RCU_TRNG_SLP	TRNG clock
RCU_TMU_SLP	TMU clock
RCU_RAMECCMU1_SLP	RAMECCMU1 clock
RCU_EXMC_SLP	EXMC clock
RCU_MDMA_SLP	MDMA clock
RCU_OSPIM_SLP	OSPIM clock
RCU_OSPI0_SLP	OSPI0 clock
RCU_OSPI1_SLP	OSPI1 clock
RCU_RTDEC0_SLP	RTDEC0 clock
RCU_RTDEC1_SLP	RTDEC1 clock
RCU_RAMECCMU0_SLP	RAMECCMU0 clock
RCU_AXISRAM_SLP	AXISRAM clock
RCU_FMC_SLP	FMC clock
RCU_GPIOA_SLP	GPIOA clock
RCU_GPIOB_SLP	GPIOB clock
RCU_GPIOC_SLP	GPIOC clock
RCU_GPIOD_SLP	GPIOD clock
RCU_GPIOE_SLP	GPIOE clock
RCU_GPIOF_SLP	GPIOF clock
RCU_GPIOG_SLP	GPIOG clock
RCU_GPIOH_SLP	GPIOH clock
RCU_GPIOJ_SLP	GPIOJ clock
RCU_GPIOK_SLP	GPIOK clock
RCU_BKPSRAM_SLP	BKPSRAM clock
RCU_CRC_SLP	CRC clock

enum name	Function description
RCU_TIMER1_SLP	TIMER1 clock
RCU_TIMER2_SLP	TIMER2 clock
RCU_TIMER3_SLP	TIMER3 clock
RCU_TIMER4_SLP	TIMER4 clock
RCU_TIMER5_SLP	TIMER5 clock
RCU_TIMER6_SLP	TIMER6 clock
RCU_TIMER22_SLP	TIMER22 clock
RCU_TIMER23_SLP	TIMER23 clock
RCU_TIMER50_SLP	TIMER50 clock
RCU_TIMER51_SLP	TIMER51 clock
RCU_SPI1_SLP	SPI1 clock
RCU_SPI2_SLP	SPI2 clock
RCU_USART1_SLP	USART1 clock
RCU_USART2_SLP	USART2 clock
RCU_UART3_SLP	UART3 clock
RCU_UART4_SLP	UART4 clock
RCU_I2C0_SLP	I2C0 clock
RCU_I2C1_SLP	I2C1 clock
RCU_I2C2_SLP	I2C2 clock
RCU_I2C3_SLP	I2C3 clock
RCU_CTC_SLP	CTC clock
RCU_DACHOLD_SLP	DACHOLD clock
RCU_DAC_SLP	DAC clock
RCU_UART6_SLP	UART6 clock
RCU_UART7_SLP	UART7 clock
RCU_TIMER0_SLP	TIMER0 clock
RCU_TIMER7_SLP	TIMER7 clock
RCU_USART0_SLP	USART0 clock
RCU_USART5_SLP	USART5 clock
RCU_ADC0_SLP	ADC0 clock
RCU_ADC1_SLP	ADC1 clock
RCU_ADC2_SLP	ADC2 clock
RCU_SPI0_SLP	SPI0 clock
RCU_SPI3_SLP	SPI3 clock
RCU_TIMER14_SLP	TIMER14 clock
RCU_TIMER15_SLP	TIMER15 clock
RCU_TIMER16_SLP	TIMER16 clock
RCU_HPDF_SLP	HPDF clock
RCU_SPI4_SLP	SPI4 clock
RCU_SPI5_SLP	SPI5 clock
RCU_TIMER40_SLP	TIMER40 clock
RCU_TIMER41_SLP	TIMER41 clock

enum name	Function description
RCU_TIMER42_SLP	TIMER42 clock
RCU_TIMER43_SLP	TIMER43 clock
RCU_TIMER44_SLP	TIMER44 clock
RCU_EDOUT_SLP	EDOUT clock
RCU_TRIGSEL_SLP	TRIGSEL clock
RCU_WWDGT_SLP	WWDGT clock
RCU_SYSCFG_SLP	SYSCFG clock
RCU_CMP_SLP	CMP clock
RCU_VREF_SLP	VREF clock
RCU_LPPTS_SLP	LPPTS clock
RCU_PMU_SLP	PMU clock
RCU_CAN0_SLP	CAN0 clock
RCU_CAN1_SLP	CAN1 clock
RCU_CAN2_SLP	CAN2 clock

### Enum rcu\_periph\_reset\_enum

**Table 3-898. Enum rcu\_periph\_reset\_enum**

enum name	Function description
RCU_DMA0RST	DMA0 clock reset
RCU_DMA1RST	DMA1 clock reset
RCU_DMAMUXRST	DMAMUX clock reset
RCU_ENET0RST	ENET clock reset
RCU_USBHS1RST	USBHS1HS clock reset
RCU_DCIRST	DCI clock reset
RCU_FACRST	FAC clock reset
RCU_TRNGRST	TRNG clock reset
RCU_TMURST	TMU clock reset
RCU_EXMCRST	EXMC clock reset
RCU_MDMARST	MDMMA clock reset
RCU_OSPIMRST	OSPIM clock reset
RCU_OSPI0RST	OSPI0 clock reset
RCU_OSPI1RST	OSPI1 clock reset
RCU_RTDEC0RST	RTDEC0 clock reset
RCU_RTDEC1RST	RTDEC1 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

enum name	Function description
RCU_GPIOHRST	GPIOH clock reset
RCU_GPIOJRST	GPIOJ clock reset
RCU_GPIOKRST	GPIOK clock reset
RCU_CRCRST	CRC clock reset
RCU_TIMER1RST	TIMER1 clock reset
RCU_TIMER2RST	TIMER2 clock reset
RCU_TIMER3RST	TIMER3 clock reset
RCU_TIMER4RST	TIMER4 clock reset
RCU_TIMER5RST	TIMER5 clock reset
RCU_TIMER6RST	TIMER6 clock reset
RCU_TIMER22RST	TIMER22 clock reset
RCU_TIMER23RST	TIMER23 clock reset
RCU_TIMER50RST	TIMER50 clock reset
RCU_TIMER51RST	TIMER51 clock reset
RCU_SPI1RST	SPI1 clock reset
RCU_SPI2RST	SPI2 clock reset
RCU_USART1RST	USART1 clock reset
RCU_USART2RST	USART2 clock reset
RCU_UART3RST	UART3 clock reset
RCU_UART4RST	UART4 clock reset
RCU_I2C0RST	I2C0 clock reset
RCU_I2C1RST	I2C1 clock reset
RCU_I2C2RST	I2C2 clock reset
RCU_I2C3RST	I2C3 clock reset
RCU_CTCRST	CTC clock reset
RCU_DACHOLDRST	DACHOLD clock reset
RCU_DACRST	DAC clock reset
RCU_UART6RST	UART6 clock reset
RCU_UART7RST	UART7 clock reset
RCU_TIMER0RST	TIMER0 clock reset
RCU_TIMER7RST	TIMER7 clock reset
RCU_USART0RST	USART0 clock reset
RCU_USART5RST	USART5 clock reset
RCU_ADC0RST	ADC0 clock reset
RCU_ADC1RST	ADC1 clock reset
RCU_ADC2RST	ADC2 clock reset
RCU_SPI0RST	SPI0 clock reset
RCU_SPI3RST	SPI3 clock reset
RCU_TIMER14RST	TIMER14 clock reset
RCU_TIMER15RST	TIMER15 clock reset
RCU_TIMER16RST	TIMER16 clock reset
RCU_HPDRST	HPDF clock reset

enum name	Function description
RCU_SPI4RST	SPI4 clock reset
RCU_SPI5RST	SPI5 clock reset
RCU_TIMER40RST	TIMER40 clock reset
RCU_TIMER41RST	TIMER41 clock reset
RCU_TIMER42RST	TIMER42 clock reset
RCU_TIMER43RST	TIMER43 clock reset
RCU_TIMER44RST	TIMER44 clock reset
RCU_EDOUTRST	EDOUT clock reset
RCU_TRIGSELRST	TRIGSEL clock reset
RCU_WWDGTRST	WWDGT clock reset
RCU_SYSCFGRST	SYSCFG clock reset
RCU_CMPRST	CMP clock reset
RCU_VREFRST	VREF clock reset
RCU_LPDTSRST	LPDTS clock reset
RCU_PMURST	PMU clock reset
RCU_CAN0RST	CAN0 clock reset
RCU_CAN1RST	CAN1 clock reset
RCU_CAN2RST	CAN2 clock reset

## Enum rcu\_flag\_enum

**Table 3-899. Enum rcu\_flag\_enum**

enum name	Function description
RCU_FLAG_IRC64MS TB	IRC64M stabilization flag
RCU_FLAG_HXTALST B	HXTAL stabilization flag
RCU_FLAG_PLL0STB	PLL0 stabilization flag
RCU_FLAG_PLL1STB	PLL1 stabilization flag
RCU_FLAG_PLL2STB	PLL2 stabilization flag
RCU_FLAG_LXTALST B	LXTAL stabilization flag
RCU_FLAG_IRC32KST B	IRC32K stabilization flags
RCU_FLAG_IRC48MS TB	IRC48M stabilization flags
RCU_FLAG_LPIRC4M STB	LPIRC4M stabilization flags
RCU_FLAG_PLLUSBH S0STB	PLLUSBHS0 stabilization flags
RCU_FLAG_PLLUSBH S1STB	PLLUSBHS1 stabilization flags
RCU_FLAG_LCKMD	LXTAL clock failure detection flags

enum name	Function description
RCU_FLAG BORRST	BOR reset flags
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 flags

### Enum rcu\_int\_flag\_enum

**Table 3-900. Enum rcu\_int\_flag\_enum**

enum name	Function description
RCU_INT_FLAG_IRC3 2KSTB	IRC32K stabilization interrupt flag
RCU_INT_FLAG_LXTA LSTB	LXTAL stabilization interrupt flag
RCU_INT_FLAG_IRC6 4MSTB	IRC64M stabilization interrupt flag
RCU_INT_FLAG_HXTA LSTB	HXTAL stabilization interrupt flag
RCU_INT_FLAG_PLL0 STB	PLL0 stabilization interrupt flag
RCU_INT_FLAG_PLL1 STB	PLL1 stabilization interrupt flag
RCU_INT_FLAG_PLL2 STB	PLL2 stabilization interrupt flag
RCU_INT_FLAG_CKM	HXTAL clock monitor interrupt flag
RCU_INT_FLAG_LCK M	LXTAL clock monitor interrupt flag
RCU_INT_FLAG_LPIR C4MSTB	LPIRC4M stabilization interrupt flag
RCU_INT_FLAG_IRC4 8MSTB	IRC48M stabilization interrupt flag
RCU_INT_FLAG_PLLU SBHS0STB	PLLUSBHS0 stabilization interrupt flag
RCU_INT_FLAG_PLLU SBHS1STB	PLLUSBHS1 stabilization interrupt flag

## Enum rcu\_int\_flag\_clear\_enum

**Table 3-901. Enum rcu\_int\_flag\_clear\_enum**

enum name	Function description
RCU_INT_FLAG_IRC3 2KSTB_CLR	IRC32K stabilization interrupt flag clear
RCU_INT_FLAG_LXTA LSTB_CLR	LXTAL stabilization interrupt flag clear
RCU_INT_FLAG_IRC6 4MSTB_CLR	IRC64M stabilization interrupt flag clear
RCU_INT_FLAG_HXTA LSTB_CLR	HXTAL stabilization interrupt flag clear
RCU_INT_FLAG_PLL0 STB_CLR	PLL0 stabilization interrupt flag clear
RCU_INT_FLAG_PLL1 STB_CLR	PLL1 stabilization interrupt flag clear
RCU_INT_FLAG_PLL2 STB_CLR	PLL2 stabilization interrupt flag clear
RCU_INT_FLAG_CKM _CLR	HXTAL clock stuck interrupt flags clear
RCU_INT_FLAG_LCK M_CLR	LXTAL clock stuck interrupt flags clear
RCU_INT_FLAG_LPIR C4MSTB_CLR	LPIRC4M stabilization interrupt flag clear
RCU_INT_FLAG_IRC4 8MSTB_CLR	IRC48M stabilization interrupt flag clear
RCU_INT_FLAG_PLLU SBHS0STB_CLR	PLLUSBHS0 stabilization interrupt flag clear
RCU_INT_FLAG_PLLU SBHS1STB_CLR	PLLUSBHS1 stabilization interrupt flag clear

## Enum rcu\_int\_enum

**Table 3-902. Enum rcu\_int\_enum**

enum name	Function description
RCU_INT_IRC32KSTB	IRC32K stabilization interrupt
RCU_INT_LXTALSTB	LXTAL stabilization interrupt
RCU_INT_IRC64MSTB	IRC64M stabilization interrupt
RCU_INT_HXTALSTB	HXTAL stabilization interrupt
RCU_INT_PLL0STB	PLL0 stabilization interrupt
RCU_INT_PLL1STB	PLL1 stabilization interrupt
RCU_INT_PLL2STB	PLL2 stabilization interrupt
RCU_INT_IRC48MSTB	internal 48 MHz RC oscillator stabilization interrupt
RCU_INT_PLLUSBHS0	PLLUSBHS0 stabilization interrupt



enum name	Function description
STB	
RCU_INT_PLLUSBHS1 STB	PLLUSBHS1 stabilization interrupt

### Enum rcu\_osci\_type\_enum

**Table 3-903. Enum rcu\_osci\_type\_enum**

enum name	Function description
RCU_HXTAL	HXTAL
RCU_LXTAL	LXTAL
RCU_IRC64M	IRC64M
RCU_IRC48M	IRC48M
RCU_IRC32K	IRC32K
RCU_LPIRC4M	LPIRC4M
RCU_PLL0_CK	PLL0
RCU_PLL1_CK	PLL1
RCU_PLL2_CK	PLL2
RCU_PLLUSBHS0_CK	PLLUSBHS0
RCU_PLLUSBHS1_CK	PLLUSBHS1

### Enum rcu\_clock\_freq\_enum

**Table 3-904. Enum rcu\_clock\_freq\_enum**

enum name	Function description
CK_SYS	system clock
CK_AHB	AHB clock
CK_APB1	APB1 clock
CK_APB2	APB2 clock
CK_APB3	APB3 clock
CK_APB4	APB4 clock
CK_PLL0P	PLL0P clock
CK_PLL0Q	PLL0Q clock
CK_PLL0R	PLL0R clock
CK_PLL1P	PLL1P clock
CK_PLL1Q	PLL1Q clock
CK_PLL1R	PLL1R clock
CK_PLL2P	PLL2P clock
CK_PLL2Q	PLL2Q clock
CK_PLL2R	PLL2R clock
CK_PER	PER clock
CK_USART0	USART0 clock
CK_USART1	USART1 clock
CK_USART2	USART2 clock

enum name	Function description
CK_USART5	USART5 clock
CK_IRC64MDIV	IRC64MDIV clock
CK_HXTAL	HXTAL clock
CK_LPIRC4M	LPIRC4M clock

### Enum usart\_idx\_enum

**Table 3-905. Enum usart\_idx\_enum**

enum name	Function description
IDX_USART0	idnex of USART0
IDX_USART1	idnex of USART1
IDX_USART2	idnex of USART2
IDX_USART5	idnex of USART5

### Enum i2c\_idx\_enum

**Table 3-906. Enum i2c\_idx\_enum**

enum name	Function description
IDX_I2C0	idnex of I2C0
IDX_I2C1	idnex of I2C1
IDX_I2C2	idnex of I2C2
IDX_I2C3	idnex of I2C3

### Enum can\_idx\_enum

**Table 3-907. Enum can\_idx\_enum**

enum name	Function description
IDX_CAN0	idnex of CAN0
IDX_CAN1	idnex of CAN1
IDX_CAN2	idnex of CAN2

### Enum adc\_idx\_enum

**Table 3-908. Enum adc\_idx\_enum**

enum name	Function description
IDX_ADC0	idnex of ADC0
IDX_ADC1	idnex of ADC1
IDX_ADC2	idnex of ADC2

### Enum usbhs\_idx\_enum

**Table 3-909. Enum usbhs\_idx\_enum**

enum name	Function description
IDX_USBHS0	idnex of USBHS0

enum name	Function description
IDX_USBHS1	idnex of USBHS1

### Enum pll\_idx\_enum

**Table 3-910. Enum pll\_idx\_enum**

enum name	Function description
IDX_PLL0	idnex of PLL0
IDX_PLL1	idnex of PLL1
IDX_PLL2	idnex of PLL2

### Enum spi\_idx\_enum

**Table 3-911. Enum spi\_idx\_enum**

enum name	Function description
IDX_SPI0	idnex of SPI0
IDX_SPI1	idnex of SPI1
IDX_SPI2	idnex of SPI2
IDX_SPI3	idnex of SPI3
IDX_SPI4	idnex of SPI4
IDX_SPI5	idnex of SPI5

### rcu\_deinit

The description of rcu\_deinit is shown as below:

**Table 3-912. 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 3-913. 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 3-896. Enum rcu_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## rcu\_periph\_clock\_disable

The description of rcu\_periph\_clock\_disable is shown as below:

**Table 3-914. Function rcu\_periph\_clock\_disable**

<b>Function name</b>	rcu_periph_clock_disable
<b>Function prototype</b>	void rcu_periph_clock_disable(rcu_periph_enum periph);
<b>Function descriptions</b>	disable the peripherals clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>periph</b>	RCU peripherals, refer to <a href="#">Table 3-896. Enum rcu_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## rcu\_periph\_clock\_sleep\_enable

The description of rcu\_periph\_clock\_sleep\_enable is shown as below:

**Table 3-915. Function rcu\_periph\_clock\_sleep\_enable**

<b>Function name</b>	rcu_periph_clock_sleep_enable
<b>Function prototype</b>	void rcu_periph_clock_sleep_enable(rcu_periph_sleep_enum periph);
<b>Function descriptions</b>	enable the peripherals clock when in sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>periph</b>	RCU peripherals, refer to <a href="#">Table 3-897. Enum rcu_periph_sleep_enum</a>
<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 3-916. 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 in 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 3-897. Enum rcu_periph_sleep_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### rcu\_periph\_reset\_enable

The description of rcu\_periph\_reset\_enable is shown as below:

**Table 3-917. 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>	enable the peripherals reset
<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 3-898. Enum rcu_periph_reset_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### rcu\_periph\_reset\_disable

The description of rcu\_periph\_reset\_disable is shown as below:

**Table 3-918. 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 the peripheral reset
<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 3-898. Enum rcu_periph_reset_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

### rcu\_bkp\_reset\_enable

The description of rcu\_bkp\_reset\_enable is shown as below:

**Table 3-919. Function rcu\_bkp\_reset\_enable**

<b>Function name</b>	rcu_bkp_reset_enable
----------------------	----------------------

<b>Function prototype</b>	void rcu_bkp_reset_enable(void);
<b>Function descriptions</b>	enable the BKP domain reset
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset the BKP domain */
rcu_bkp_reset_enable();
```

### rcu\_bkp\_reset\_disable

The description of rcu\_bkp\_reset\_disable is shown as below:

**Table 3-920. Function rcu\_bkp\_reset\_disable**

<b>Function name</b>	rcu_bkp_reset_disable
<b>Function prototype</b>	void rcu_bkp_reset_disable(void);
<b>Function descriptions</b>	disable the BKP domain reset
<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 BKP domain reset */
rcu_bkp_reset_disable();
```

### rcu\_system\_clock\_source\_config

The description of rcu\_system\_clock\_source\_config is shown as below:

**Table 3-921. 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_IRC64MDIV</i>	select CK_IRC64MDIV as the CK_SYS source
<i>RCU_CKSYSSRC_HXTAL</i>	select CK_HXTAL as the CK_SYS source
<i>RCU_CKSYSSRC_LPIRC4M</i>	select CK_LPIRC4M as the CK_SYS source
<i>RCU_CKSYSSRC_PLL0P</i>	select CK_PLL0P 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 3-922. Function rcu\_system\_clock\_source\_get**

<b>Function name</b>	rcu_system_clock_source_get
<b>Function prototype</b>	uint32_t rcu_system_clock_source_get(void);
<b>Function descriptions</b>	get the system clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	RCU_SCSS_IRC64MDIV / RCU_SCSS_HXTAL / RCU_SCSS_LPIRC4M / RCU_SCSS_PLL0P

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 3-923. Function rcu\_ahb\_clock\_config**

<b>Function name</b>	rcu_ahb_clock_config
<b>Function prototype</b>	void rcu_ahb_clock_config(uint32_t ck_ahb);
<b>Function descriptions</b>	configure the AHB clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_ahb</b>	AHB clock prescaler selection
<i>RCU_AHB_CKSYS_DIVx</i>	select CK_SYS / x, (x = 1, 2, 4, 8, 16, 64, 128, 256, 512)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CK_SYS/128 */
```

```
rcu_ahb_clock_config(RCU_AHB_CKSYS_DIV128);
```

## rcu\_apb1\_clock\_config

The description of rcu\_apb1\_clock\_config is shown as below:

**Table 3-924. Function rcu\_apb1\_clock\_config**

<b>Function name</b>	rcu_apb1_clock_config
<b>Function prototype</b>	void rcu_apb1_clock_config(uint32_t ck_apb1);
<b>Function descriptions</b>	configure the APB1 clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_apb1</b>	APB1 clock prescaler selection
<i>RCU_APB1_CKAHB_DIVx</i>	select (CK_AHB / x) as CK_APB1 (x = 1, 2, 4, 8, 16)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CK_AHB/16 as CK_APB1 */
```

```
rcu_apb1_clock_config(RCU_APB1_CKAHB_DIV16);
```

### rcu\_apb2\_clock\_config

The description of rcu\_apb2\_clock\_config is shown as below:

**Table 3-925. Function rcu\_apb2\_clock\_config**

<b>Function name</b>	rcu_apb2_clock_config
<b>Function prototype</b>	void rcu_apb2_clock_config(uint32_t ck_apb2);
<b>Function descriptions</b>	configure the APB2 clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_apb2</b>	APB2 clock prescaler selection
<i>RCU_APB2_CKAHB_D</i> <i>IVx</i>	select (CK_AHB / x) as CK_APB2 clock (x = 1, 2, 4, 8, 16)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB2 */
```

```
rcu_apb2_clock_config(RCU_APB2_CKAHB_DIV8);
```

### rcu\_apb3\_clock\_config

The description of rcu\_apb3\_clock\_config is shown as below:

**Table 3-926. Function rcu\_apb3\_clock\_config**

<b>Function name</b>	rcu_apb3_clock_config
<b>Function prototype</b>	void rcu_apb3_clock_config(uint32_t ck_apb3);
<b>Function descriptions</b>	configure the APB3 clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_apb3</b>	APB3 clock prescaler selection
<i>RCU_APB3_CKAHB_D</i> <i>IVx</i>	select (CK_AHB / x) as CK_APB3 clock (x = 1, 2, 4, 8, 16)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB3 */
```

```
rcu_apb3_clock_config(RCU_APB3_CKAHB_DIV8);
```

### rcu\_apb4\_clock\_config

The description of rcu\_apb4\_clock\_config is shown as below:

**Table 3-927. Function rcu\_apb4\_clock\_config**

<b>Function name</b>	rcu_apb4_clock_config
<b>Function prototype</b>	void rcu_apb4_clock_config(uint32_t ck_apb4);
<b>Function descriptions</b>	configure the APB4 clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_apb4</b>	APB4 clock prescaler selection
<i>RCU_APB4_CKAHB_DIVx</i>	select (CK_AHB / x) as CK_APB4 clock (x = 1, 2, 4, 8, 16)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure CK_AHB/8 as CK_APB4 */
```

```
rcu_apb4_clock_config(RCU_APB4_CKAHB_DIV8);
```

### rcu\_ckout0\_config

The description of rcu\_ckout0\_config is shown as below:

**Table 3-928. Function rcu\_ckout0\_config**

<b>Function name</b>	rcu_ckout0_config
<b>Function prototype</b>	void rcu_ckout0_config(uint32_t ckout0_src, uint32_t ckout0_div);
<b>Function descriptions</b>	configure the CK_OUT0 clock source and divider
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ckout0_src</b>	CK_OUT0 clock source selection
<i>RCU_CKOUT0SRC_IRC64MDIV</i>	IRC64M selected
<i>RCU_CKOUT0SRC_LXTAL</i>	LXTAL selected
<i>RCU_CKOUT0SRC_HXTAL</i>	HXTAL selected

<i>RCU_CKOUT0SRC_PL L0P</i>	PLL0P selected
<i>RCU_CKOUT0SRC_IR C48M</i>	IRC48M selected
<i>RCU_CKOUT0SRC_P ER</i>	PER selected
<i>RCU_CKOUT0SRC_U SBHS060M</i>	USBHS0 60M selected
<i>RCU_CKOUT0SRC_U SBHS160M</i>	USBHS1 60M selected
<b>ckout0_div</b>	CK_OUT0 divider
<i>RCU_CKOUT0_DIVx</i>	CK_OUT is divided by x(x = 1, 2, 3...15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the HXTAL as CK_OUT0 clock source */
```

```
rcu_ckout0_config(RCU_CKOUT0SRC_HXTAL, RCU_CKOUT0_DIV1);
```

### rcu\_ckout1\_config

The description of rcu\_ckout1\_config is shown as below:

**Table 3-929. Function rcu\_ckout1\_config**

<b>Function name</b>	rcu_ckout1_config
<b>Function prototype</b>	void rcu_ckout1_config(uint32_t ckout1_src, uint32_t ckout1_div);
<b>Function descriptions</b>	configure the CK_OUT1 clock source and divider
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ckout1_src</b>	CK_OUT1 clock source selection
<i>RCU_CKOUT1SRC_S YSTEMCLOCK</i>	system clock selected
<i>RCU_CKOUT1SRC_PL L1R</i>	PLL1R selected
<i>RCU_CKOUT1SRC_H XTAL</i>	HXTAL selected
<i>RCU_CKOUT1SRC_PL L0P</i>	PLL0P selected
<i>RCU_CKOUT1SRC_LP IRC4M</i>	LPIRC4M selected

<i>RCU_CKOUT1SRC_IR</i> <i>C32K</i>	IRC32K selected
<i>RCU_CKOUT1SRC_PL</i> <i>L2R</i>	PLL2R selected
<b>Input parameter{in}</b>	
<b>ckout1_div</b>	CK_OUT1 divider
<i>RCU_CKOUT1_DIVx</i>	CK_OUT1 is divided by x(x = 1, 2, 3...15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the HXTAL as CK_OUT1 clock source */
```

```
rcu_ckout1_config(RCU_CKOUT1SRC_HXTAL, RCU_CKOUT1_DIV1);
```

### rcu\_pll\_input\_output\_clock\_range\_config

The description of rcu\_pll\_input\_output\_clock\_range\_config is shown as below:

**Table 3-930. Function rcu\_pll\_input\_output\_clock\_range\_config**

<b>Function name</b>	rcu_pll_input_output_clock_range_config
<b>Function prototype</b>	void rcu_pll_input_output_clock_range_config(pll_idx_enum pll_idx, uint32_t ck_input, uint32_t ck_output);
<b>Function descriptions</b>	configure the pll input and output clock range
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll_idx</b>	pll index, refer to <a href="#">Table 3-910. Enum pll_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_input</b>	input clock range
<i>RCU_PLLRNG_1M_2M</i>	input clock frequency: 1-2MHz
<i>RCU_PLLRNG_2M_4M</i>	input clock frequency: 2-4MHz
<i>RCU_PLLRNG_4M_8M</i>	input clock frequency: 4-8MHz
<i>RCU_PLLRNG_8M_16M</i>	input clock frequency: 8-16MHz
<b>Input parameter{in}</b>	
<i>RCU_PLLVCO_192M_836M</i>	select wide VCO, range: 192-836MHz
<i>RCU_PLLVCO_150M_420M</i>	select narrow VCO, range: 150-420MHz
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* configure the pll0 input and output clock range */
```

```
rcu_pll_input_output_clock_range_config(IDX_PLL0, RCU_PLLRNG_4M_8M,
RCU_PLLVCO_192M_836M);
```

### rcu\_pll\_fractional\_config

The description of rcu\_pll\_fractional\_config is shown as below:

**Table 3-931. Function rcu\_pll\_fractional\_config**

<b>Function name</b>	rcu_pll_fractional_config
<b>Function prototype</b>	void rcu_pll_fractional_config(pll_idx_enum pll_idx, uint32_t pll_fracn);
<b>Function descriptions</b>	configure fractional part of the multiplication factor for PLL VCO
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll_idx</b>	pll index, refer to <a href="#">Table 3-910. Enum pll_idx_enum</a>
<b>Input parameter{in}</b>	
<b>pll_fracn</b>	fractional part of the multiplication factor
<b>uint32_t</b>	0x00-0x00001fff
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure fractional part of the multiplication factor for PLL0 VCO */
```

```
rcu_pll_fractional_config(IDX_PLL0, 1);
```

### rcu\_pll\_fractional\_latch\_enable

The description of rcu\_pll\_fractional\_latch\_enable is shown as below:

**Table 3-932. Function rcu\_pll\_fractional\_latch\_enable**

<b>Function name</b>	rcu_pll_fractional_latch_enable
<b>Function prototype</b>	void rcu_pll_fractional_latch_enable(pll_idx_enum pll_idx);
<b>Function descriptions</b>	PLL fractional latch enable
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll_idx</b>	pll index, refer to <a href="#">Table 3-910. Enum pll_idx_enum</a>
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* enable PLL0 fractional latch */
```

```
rcu_pll_fractional_latch_enable(IDX_PLL0);
```

### rcu\_pll\_fractional\_latch\_disable

The description of rcu\_pll\_fractional\_latch\_disable is shown as below:

**Table 3-933. Function rcu\_pll\_fractional\_latch\_disable**

Function name	rcu_pll_fractional_latch_disable
Function prototype	void rcu_pll_fractional_latch_disable(pll_idx_enum pll_idx);
Function descriptions	PLL fractional latch disable
Precondition	-
The called functions	-
Input parameter{in}	
pll_idx	pll index, refer to <a href="#">Table 3-910. Enum pll_idx_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable PLL0 fractional latch */
```

```
rcu_pll_fractional_latch_disable(IDX_PLL0);
```

### rcu\_pll\_source\_config

The description of rcu\_pll\_source\_config is shown as below:

**Table 3-934. Function rcu\_pll\_source\_config**

Function name	rcu_pll_source_config
Function prototype	void rcu_pll_source_config(uint32_t pll_src);
Function descriptions	configure PLL clock source
Precondition	-
The called functions	-
Input parameter{in}	
pll_src	PLL clock source selection
RCU_PLLSRC_IRC64MDIV	select IRC64MDIV as PLL source clock
RCU_PLLSRC_LPIRC4	select LPIRC4M as PLL source clock

<i>M</i>	
<i>RCU_PLLSRC_HXTAL</i>	select HXTAL as PLL source clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select RCU_PLLSRC_HXTAL as the PLL clock source */
```

```
rcu_pll_source_config(RCU_PLLSRC_HXTAL);
```

### rcu\_pll0\_config

The description of rcu\_pll0\_config is shown as below:

**Table 3-935. Function rcu\_pll0\_config**

<b>Function name</b>	rcu_pll0_config
<b>Function prototype</b>	ErrStatus rcu_pll0_config(uint32_t pll0_psc, uint32_t pll0_n, uint32_t pll0_p, uint32_t pll0_q, uint32_t pll0_r);
<b>Function descriptions</b>	configure the main PLL0 clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll0_psc</b>	the PLL0 VCO source clock prescaler
<i>uint32_t</i>	1-63
<b>Input parameter{in}</b>	
<b>pll0_n</b>	the PLL0 VCO clock multi factor
<i>uint32_t</i>	9-512
<b>Input parameter{in}</b>	
<b>pll0_p</b>	the PLL0P output frequency division factor from PLL0 VCO clock
<i>uint32_t</i>	1-128
<b>Input parameter{in}</b>	
<b>pll0_q</b>	the PLL0Q output frequency division factor from PLL0 VCO clock
<i>uint32_t</i>	1-128
<b>Input parameter{in}</b>	
<b>pll0_r</b>	the PLL0R output frequency division factor from PLL0 VCO clock
<i>uint32_t</i>	1-128
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR-

Example:

```
/* configure the PLL0 */
```



```
rcu_pll0_config(1, 200, 1, 2, 2);
```

## rcu\_pll1\_config

The description of rcu\_pll1\_config is shown as below:

**Table 3-936. Function rcu\_pll1\_config**

<b>Function name</b>	rcu_pll1_config
<b>Function prototype</b>	ErrStatus rcu_pll1_config(uint32_t pll1_psc, uint32_t pll1_n, uint32_t pll1_p, uint32_t pll1_q, uint32_t pll1_r);
<b>Function descriptions</b>	configure the PLL1 clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll1_psc</b>	the PLL1 VCO source clock prescaler
<b>uint32_t</b>	1-63
<b>Input parameter{in}</b>	
<b>pll1_n</b>	the PLL1 VCO clock multi factor
<b>uint32_t</b>	9-512
<b>Input parameter{in}</b>	
<b>pll1_p</b>	the PLL1 P output frequency division factor from PLL1 VCO clock
<b>uint32_t</b>	1-128
<b>Input parameter{in}</b>	
<b>pll1_q</b>	the PLL1 Q output frequency division factor from PLL1 VCO clock
<b>uint32_t</b>	1-128
<b>Input parameter{in}</b>	
<b>pll1_r</b>	the PLL1 R output frequency division factor from PLL1 VCO clock
<b>uint32_t</b>	1-128
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR-

Example:

```
/* configure the PLL1 */
```

```
rcu_pll1_config(1, 200, 1, 2, 2);
```

## rcu\_pll2\_config

The description of rcu\_pll2\_config is shown as below:

**Table 3-937. Function rcu\_pll2\_config**

<b>Function name</b>	rcu_pll2_config
<b>Function prototype</b>	ErrStatus rcu_pll2_config(uint32_t pll2_psc, uint32_t pll2_n, uint32_t pll2_p, uint32_t pll2_q, uint32_t pll2_r);

<b>Function descriptions</b>	configure the PLL2 clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll2_psc</b>	the PLL2 VCO source clock prescaler
<i>uint32_t</i>	1-63
<b>Input parameter{in}</b>	
<b>pll2_n</b>	the PLL2 VCO clock multi factor
<i>uint32_t</i>	9-512
<b>Input parameter{in}</b>	
<b>pll2_p</b>	the PLL2 P output frequency division factor from PLL2 VCO clock
<i>uint32_t</i>	1-128
<b>Input parameter{in}</b>	
<b>pll2_q</b>	the PLL2 Q output frequency division factor from PLL2 VCO clock
<i>uint32_t</i>	1-128
<b>Input parameter{in}</b>	
<b>pll2_r</b>	the PLL2 R output frequency division factor from PLL2 VCO clock
<i>uint32_t</i>	1-128
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR-

Example:

```
/* configure the PLL2 */
```

```
rcu_pll2_config(1, 200, 1, 2, 2);
```

### rcu\_pll\_clock\_output\_enable

The description of rcu\_pll\_clock\_output\_enable is shown as below:

**Table 3-938. Function rcu\_pll\_clock\_output\_enable**

<b>Function name</b>	rcu_pll_clock_output_enable
<b>Function prototype</b>	void rcu_pll_clock_output_enable(uint32_t pllxy);
<b>Function descriptions</b>	enable the pll p pll q pll r divider output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pllxy</b>	the output pll enable
<i>RCU_PLL0P</i>	PLL0P divider output enable
<i>RCU_PLL0Q</i>	PLL0Q divider output enable
<i>RCU_PLL0R</i>	PLL0R divider output enable

<i>RCU_PLL1P</i>	PLL1P divider output enable
<i>RCU_PLL1Q</i>	PLL1Q divider output enable
<i>RCU_PLL1R</i>	PLL1R divider output enable
<i>RCU_PLL2P</i>	PLL2P divider output enable
<i>RCU_PLL2Q</i>	PLL2Q divider output enable
<i>RCU_PLL2R</i>	PLL2R divider output enable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable PLL0P divider output */
rcu_pll_clock_output_enable(RCU_PLL0P);
```

### rcu\_pll\_clock\_output\_disable

The description of rcu\_pll\_clock\_output\_disable is shown as below:

**Table 3-939. Function rcu\_pll\_clock\_output\_disable**

<b>Function name</b>	rcu_pll_clock_output_disable
<b>Function prototype</b>	void rcu_pll_clock_output_disable(uint32_t pllxy);
<b>Function descriptions</b>	disable the pll p pllq pllr divider output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pllxy</b>	the output pll disable
<i>RCU_PLL0P</i>	PLL0P divider output disable
<i>RCU_PLL0Q</i>	PLL0Q divider output disable
<i>RCU_PLL0R</i>	PLL0R divider output disable
<i>RCU_PLL1P</i>	PLL1P divider output disable
<i>RCU_PLL1Q</i>	PLL1Q divider output disable
<i>RCU_PLL1R</i>	PLL1R divider output disable
<i>RCU_PLL2P</i>	PLL2P divider output disable
<i>RCU_PLL2Q</i>	PLL2Q divider output disable
<i>RCU_PLL2R</i>	PLL2R divider output disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable PLL0P divider output */
```

```
rcu_pll_clock_output_disable(RCU_PLL0P);
```

## rcu\_pllusb0\_config

The description of rcu\_pllusb0\_config is shown as below:

**Table 3-940. Function rcu\_pllusb0\_config**

<b>Function name</b>	rcu_pllusb0_config
<b>Function prototype</b>	void rcu_pllusb0_config(uint32_t pllusb_presel, uint32_t pllusb_predv, uint32_t pllusb_mf, uint32_t usbhsv);
<b>Function descriptions</b>	configure the PLLUSBHS0 clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pllusb_presel</b>	PLLUSBHS0PRE clock selection
<i>RCU_PLLUSBHSPRE_HXTAL</i>	PLLUSBHSPRE select HXTAL as clock
<i>RCU_PLLUSBHSPRE_IRC48M</i>	PLLUSBHSPRE select IRC48M as clock
<b>Input parameter{in}</b>	
<b>pllusb_predv</b>	the divider factor from PLLUSBHS0 clock
<i>RCU_PLLUSBHSPRE_DIVx (x= 1...15)</i>	select RCU_PLLUSBHSPRE_DIVx as PLLUSBHS clock divider factor
<b>Input parameter{in}</b>	
<b>pllusb_mf</b>	PLLUSBHS0 clock multiplication factor
<i>RCU_PLLUSBHS_MULx (x = 16,17...127)</i>	select RCU_PLLUSBHS_MULx as PLLUSBHS clock multiplication factor
<b>Input parameter{in}</b>	
<b>usbhsv</b>	the divider factor from USBHS0DV clock
<i>RCU_USBHS_DIVx(x = 2,4...16)</i>	select RCU_USBHS_DIVx as USBHSDV clock divider factor
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR-

Example:

```
/* configure the PLLUSBHS0 */
```

```
rcu_pllusb0_config(RCU_PLLUSBHSPRE_IRC48M, RCU_PLLUSBHSPRE_DIV1,
RCU_PLLUSBHS_MUL1, RCU_USBHS_DIV1);
```

## rcu\_pllusb1\_config

The description of rcu\_pllusb1\_config is shown as below:

**Table 3-941. Function rcu\_pllusb1\_config**

<b>Function name</b>	rcu_pllusb1_config
<b>Function prototype</b>	void rcu_pllusb1_config(uint32_t pllusb_presel, uint32_t pllusb_predv, uint32_t pllusb_mf, uint32_t usbhsv);
<b>Function descriptions</b>	configure the PLLUSBHS1 clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pllusb_presel</b>	PLLUSBHS1PRE clock selection
<i>RCU_PLLUSBHSPRE_HXTAL</i>	PLLUSBHSPRE select HXTAL as clock
<i>RCU_PLLUSBHSPRE_IRC48M</i>	PLLUSBHSPRE select IRC48M as clock
<b>Input parameter{in}</b>	
<b>pllusb_predv</b>	the divider factor from PLLUSBHS1 clock
<i>RCU_PLLUSBHSPRE_DIVx (x = 1...15)</i>	select RCU_PLLUSBHSPRE_DIVx as PLLUSBHS clock divider factor
<b>Input parameter{in}</b>	
<b>pllusb_mf</b>	PLLUSBHS1 clock multiplication factor
<i>RCU_PLLUSBHS_MULx (x = 16,17...127)</i>	select RCU_PLLUSBHS_MULx as PLLUSBHS clock multiplication factor
<b>Input parameter{in}</b>	
<b>usbhsv</b>	the divider factor from USBHS1DV clock
<i>RCU_USBHS_DIVx(x = 2,4...16)</i>	select RCU_USBHS_DIVx as USBHSDV clock divider factor
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR-

Example:

```
/* configure the PLLUSBHS1 */
```

```
rcu_pllusb1_config(RCU_PLLUSBHSPRE_IRC48M, RCU_PLLUSBHSPRE_DIV1,
RCU_PLLUSBHS_MUL1, RCU_USBHS_DIV1);
```

### rcu\_rtc\_clock\_config

The description of rcu\_rtc\_clock\_config is shown as below:

**Table 3-942. Function rcu\_rtc\_clock\_config**

<b>Function name</b>	rcu_rtc_clock_config
<b>Function prototype</b>	void rcu_rtc_clock_config(uint32_t rtc_clock_source);
<b>Function descriptions</b>	configure the RTC clock source selection
<b>Precondition</b>	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_clock_source</b>	RTC clock source selection
<i>RCU_RTCSRC_NONE</i>	no clock selected
<i>RCU_RTCSRC_LXTAL</i>	CK_LXTAL selected as RTC source clock
<i>RCU_RTCSRC_IRC32K</i>	CK_IRC32K selected as RTC source clock
<i>RCU_RTCSRC_HXTAL_DIV_RTCDIV</i>	CK_HXTAL/RTCDIV selected as RTC source clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the RTC clock source selection */
rcu_rtc_clock_config(RCU_RTCSRC_IRC32K);
```

### rcu\_rtc\_div\_config

The description of rcu\_rtc\_div\_config is shown as below:

**Table 3-943. Function rcu\_rtc\_div\_config**

<b>Function name</b>	rcu_rtc_div_config
<b>Function prototype</b>	void rcu_rtc_div_config(uint32_t rtc_div);
<b>Function descriptions</b>	configure the frequency division of RTC clock when HXTAL was selected as its clock source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_div</b>	RTC clock frequency division
<i>RCU_RTC_HXTAL_NONE</i>	no clock for RTC
<i>RCU_RTC_HXTAL_DIVx(x = 2,3...63)</i>	RTCDIV clock select CK_HXTAL/x
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select CK_HXTAL / 25 as the RTC clock source */
rcu_rtc_div_config(RCU_RTC_HXTAL_DIV25);
```

## rcu\_ck48m\_clock\_config

The description of rcu\_ck48m\_clock\_config is shown as below:

**Table 3-944. Function rcu\_ck48m\_clock\_config**

<b>Function name</b>	rcu_ck48m_clock_config
<b>Function prototype</b>	void rcu_ck48m_clock_config(uint32_t ck48m_clock_source);
<b>Function descriptions</b>	configure the CK48M clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck48m_clock_source</b>	CK48M clock source selection
<i>RCU_CK48MSRC_PLL48M</i>	CK_PLL48M selected as CK48M source clock
<i>RCU_CK48MSRC_IRC48M</i>	CK_IRC48M selected as CK48M source clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK48M clock source selection */
```

```
rcu_ck48m_clock_config(RCU_CK48MSRC_IRC48M);
```

## rcu\_pll48m\_clock\_config

The description of rcu\_pll48m\_clock\_config is shown as below:

**Table 3-945. Function rcu\_pll48m\_clock\_config**

<b>Function name</b>	rcu_pll48m_clock_config
<b>Function prototype</b>	void rcu_pll48m_clock_config(uint32_t pll48m_clock_source);
<b>Function descriptions</b>	configure the PLL48M clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>pll48m_clock_source</b>	PLL48M clock source selection
<i>RCU_PLL48MSRC_PL_L0Q</i>	CK_PLL0Q selected as PLL48M source clock
<i>RCU_PLL48MSRC_PL_L2P</i>	CK_PLL2P selected as PLL48M source clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* configure the PLL48M clock selection */
```

```
rcu_pll48m_clock_config(RCU_PLL48MSRC_PLL0Q);
```

### rcu\_irc64mdiv\_clock\_config

The description of rcu\_irc64mdiv\_clock\_config is shown as below:

**Table 3-946. Function rcu\_irc64mdiv\_clock\_config**

<b>Function name</b>	rcu_irc64mdiv_clock_config
<b>Function prototype</b>	void rcu_irc64mdiv_clock_config(uint32_t ck_irc64mdiv);
<b>Function descriptions</b>	configure the IRC64M clock divider selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_irc64mdiv</b>	IRC64M clock divider selection
RCU_IRC64M_DIV1	CK_IRC64M / 1
RCU_IRC64M_DIV2	CK_IRC64M / 2
RCU_IRC64M_DIV4	CK_IRC64M / 4
RCU_IRC64M_DIV8	CK_IRC64M / 8
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the IRC64M clock divider selection */
```

```
rcu_irc64mdiv_clock_config(RCU_IRC64M_DIV1);
```

### rcu\_irc64mdiv\_freq\_get

The description of rcu\_irc64mdiv\_freq\_get is shown as below:

**Table 3-947. Function rcu\_irc64mdiv\_freq\_get**

<b>Function name</b>	rcu_irc64mdiv_freq_get
<b>Function prototype</b>	uint32_t rcu_irc64mdiv_freq_get(void);
<b>Function descriptions</b>	get the irc64mdiv clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	



-	-
<b>Return value</b>	
uint32_t	0x0-0xFFFFFFFF

Example:

```
/* get the irc64mdiv clock */
rcu_irc64mdiv_freq_get();
```

### rcu\_timer\_clock\_prescaler\_config

The description of rcu\_timer\_clock\_prescaler\_config is shown as below:

**Table 3-948. Function rcu\_timer\_clock\_prescaler\_config**

<b>Function name</b>	rcu_timer_clock_prescaler_config
<b>Function prototype</b>	void rcu_timer_clock_prescaler_config(uint32_t timer_clock_prescaler);
<b>Function descriptions</b>	configure the TIMER clock prescaler selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_clock_prescaler</b>	TIMER clock selection
<i>RCU_TIMER_PSC_MUL2</i>	If CK_APBx = CK_AHB or CK_APBx = CK_AHB/2, CK_TIMERx = CK_AHB, else CK_TIMERx = 2 x CK_APBx
<i>RCU_TIMER_PSC_MUL4</i>	If CK_APBx = CK_AHB or CK_APBx = CK_AHB/2 or CK_APBx = CK_AHB/4, CK_TIMERx = CK_AHB, else CK_TIMERx = 4 x CK_APBx
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER clock source */
rcu_timer_clock_prescaler_config(RCU_TIMER_PSC_MUL4);
```

### rcu\_spi\_clock\_config

The description of rcu\_spi\_clock\_config is shown as below:

**Table 3-949. Function rcu\_spi\_clock\_config**

<b>Function name</b>	rcu_spi_clock_config
<b>Function prototype</b>	void rcu_spi_clock_config(spi_idx_enum spi_idx, uint32_t ck_spi);
<b>Function descriptions</b>	configure the SPI / I2S clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>spi_idx</b>	SPI index, refer to <a href="#">Table 3-911. Enum spi_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_spi</b>	SPI clock source selection
<i>RCU_SPISRC_PLL0Q</i>	select CK_PLLQ as SPI clock, for SPI0 / SPI1 / SPI2
<i>RCU_SPISRC_PLL1P</i>	select CK_PLL1P as SPI clock, for SPI0 / SPI1 / SPI2
<i>RCU_SPISRC_PLL2P</i>	select CK_PLL2P as SPI clock, for SPI0 / SPI1 / SPI2
<i>RCU_SPISRC_I2S_CKIN</i>	select I2S_CKIN as SPI clock, for SPI0 / SPI1 / SPI2
<i>RCU_SPISRC_PER</i>	select CK_PER as SPI clock, for SPI0 / SPI1 / SPI2
<i>RCU_SPISRC_APB2</i>	select CK_APB2 as SPI clock, for SPI3 / SPI4 / SPI5
<i>RCU_SPISRC_PLL1Q</i>	select CK_PLL1Q as SPI clock, for SPI3 / SPI4 / SPI5
<i>RCU_SPISRC_PLL2Q</i>	select CK_PLL2Q as SPI clock, for SPI3 / SPI4 / SPI5
<i>RCU_SPISRC_IRC64MDIV</i>	select CK_IRC64MDIV as SPI clock, for SPI3 / SPI4 / SPI5
<i>RCU_SPISRC_LPIRC4M</i>	select CK_LPIRC4M as SPI clock, for SPI3 / SPI4 / SPI5
<i>RCU_SPISRC_HXTAL</i>	select CK_HXTAL as SPI clock, for SPI3 / SPI4 / SPI5
<i>RCU_SPI5SRC_I2S_CKIN</i>	select I2S_CKIN as SPI clock, for SPI5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the PLL0Q as SPI0 clock */
rcu_spi_clock_config(IDX_SPI0, RCU_SPISRC_PLL0Q);
```

### rcu\_deepsleep\_wakeup\_sys\_clock\_config

The description of rcu\_deepsleep\_wakeup\_sys\_clock\_config is shown as below:

**Table 3-950. Function rcu\_deepsleep\_wakeup\_sys\_clock\_config**

<b>Function name</b>	rcu_deepsleep_wakeup_sys_clock_config
<b>Function prototype</b>	void rcu_deepsleep_wakeup_sys_clock_config(uint32_t ck_dspwussel);
<b>Function descriptions</b>	configure the Deep-sleep wakeup system clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_dspwussel</b>	Deep-sleep wakeup system clock source selection
<i>RCU_DSPWUSSEL_IRC64MDIV</i>	ck_dspwussel select IRC64MDIV
<i>RCU_DSPWUSSEL_LPIRC4M</i>	ck_dspwussel select LPIRC4M
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* configure the CK_LPIRC4M as Deep-sleep wakeup system clock source */
rcu_deepsleep_wakeup_sys_clock_config(RCU_DSPWUSSEL_LPIRC4M);
```

### rcu\_usart\_clock\_config

The description of rcu\_usart\_clock\_config is shown as below:

**Table 3-951. Function rcu\_usart\_clock\_config**

<b>Function name</b>	rcu_usart_clock_config
<b>Function prototype</b>	void rcu_usart_clock_config(usart_idx_enum usart_idx, uint32_t ck_usart);
<b>Function descriptions</b>	configure the USARTx(x=0,1,2,5) clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_idx</b>	USART index, refer to <a href="#">Table 3-905. Enum usart_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_usart</b>	USART clock source selection
<i>RCU_USARTSRC_APB</i>	select CK_APB as USART clock
<i>RCU_USARTSRC_AHB</i>	select CK_AHB as USART clock
<i>RCU_USARTSRC_LXTAL</i>	select CK_LXTAL as USART clock
<i>RCU_USARTSRC_IRC64MDIV</i>	select CK_IRC64MDIV as USART clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the LXTAL as USART0 clock */
rcu_usart_clock_config(IDX_USART0, RCU_USARTSRC_LXTAL);
```

### rcu\_i2c\_clock\_config

The description of rcu\_i2c\_clock\_config is shown as below:

**Table 3-952. Function rcu\_i2c\_clock\_config**

<b>Function name</b>	rcu_i2c_clock_config
----------------------	----------------------

<b>Function prototype</b>	void rcu_i2c_clock_config(i2c_idx_enum i2c_idx, uint32_t ck_i2c);
<b>Function descriptions</b>	configure the I2Cx(x=0,1,2,3) clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_idx</b>	I2C index, refer to <a href="#">Table 3-906. Enum i2c_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_i2c</b>	I2C clock source selection
<i>RCU_I2CSRC_APB1</i>	select CK_APB1 as I2C clock
<i>RCU_I2CSRC_PLL2R</i>	select CK_PLL2R as I2C clock
<i>RCU_I2CSRC_IRC64M DIV</i>	select CK_IRC64MDIV as I2C clock
<i>RCU_I2CSRC_LPIRC4 M</i>	select CK_LPIRC4M as I2C clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK_APB1 as I2C0 clock */
```

```
rcu_i2c_clock_config(IDX_I2C0, RCU_I2CSRC_APB1);
```

### rcu\_can\_clock\_config

The description of rcu\_i2c\_clock\_config is shown as below:

**Table 3-953. Function rcu\_can\_clock\_config**

<b>Function name</b>	rcu_can_clock_config
<b>Function prototype</b>	void rcu_can_clock_config(can_idx_enum can_idx, uint32_t ck_can);
<b>Function descriptions</b>	configure the CANx(x=0,1,2) clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>can_idx</b>	CAN index, refer to <a href="#">Table 3-907. Enum can_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_can</b>	CAN clock source selection
<i>RCU_CANSRC_HXTA L</i>	select CK_HXTAL as CAN clock
<i>RCU_CANSRC_APB2</i>	select CK_APB2 as CAN clock
<i>RCU_CANSRC_APB2_ DIV2</i>	select CK_APB2 / 2 as CAN clock
<i>RCU_CANSRC_IRC64 MDIV</i>	select CK_IRC64MDIV as CAN clock

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_IRC64MDIV as the CAN0 clock source */
```

```
rcu_can_clock_config(IDX_CAN0, RCU_CANSRC_IRC64MDIV);
```

### rcu\_adc\_clock\_config

The description of rcu\_adc\_clock\_config is shown as below:

**Table 3-954. Function rcu\_adc\_clock\_config**

Function name	rcu_adc_clock_config
Function prototype	void rcu_adc_clock_config(adc_idx_enum adc_idx, uint32_t ck_adc);
Function descriptions	configure the ADCx(x=0,1,2) clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
adc_idx	ADC index, refer to <a href="#">Table 3-908. Enum adc_idx_enum</a>
Input parameter{in}	
ck_adc	ADC clock source selection
RCU_ADCSRC_PLL1P	select CK_PLL1P as ADC clock
RCU_ADCSRC_PLL2R	select CK_PLL2R as ADC clock
RCU_ADCSRC_PER	select CK_PER as ADC clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_PLL1P as the ADC0 clock source */
```

```
rcu_adc_clock_config(IDX_ADC0, CK_PLL1P);
```

### rcu\_exmc\_clock\_config

The description of rcu\_exmc\_clock\_config is shown as below:

**Table 3-955. Function rcu\_exmc\_clock\_config**

Function name	rcu_exmc_clock_config
Function prototype	void rcu_exmc_clock_config(uint32_t ck_exmc);
Function descriptions	configure the EXMC clock source selection
Precondition	-

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_exmc</b>	EXMC clock source selection
<i>RCU_EXMCSRC_AHB</i>	select CK_AHB as EXMC clock
<i>RCU_EXMCSRC_PLL0</i> Q	select CK_PLL0Q as EXMC clock
<i>RCU_EXMCSRC_PLL1</i> R	select CK_PLL1R as EXMC clock
<i>RCU_EXMCSRC_PER</i>	select CK_PER as EXMC clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK_AHB as the EXMC clock source */
rcu_exmc_clock_config(RCU_EXMCSRC_AHB);
```

### rcu\_hpdf\_clock\_config

The description of rcu\_hpdf\_clock\_config is shown as below:

**Table 3-956. Function rcu\_hpdf\_clock\_config**

<b>Function name</b>	rcu_hpdf_clock_config
<b>Function prototype</b>	void rcu_hpdf_clock_config(uint32_t ck_hpdf);
<b>Function descriptions</b>	configure the HPDF clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ck_hpdf</b>	HPDF clock source selection
<i>RCU_HPDFSRC_APB2</i>	select CK_APB2 as HPDF clock
<i>RCU_HPDFSRC_AHB</i>	select CK_AHB as HPDF clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the CK_AHB as the HPDF clock source */
rcu_hpdf_clock_config(CK_AHB);
```

### rcu\_per\_clock\_config

The description of rcu\_per\_clock\_config is shown as below:

Table 3-957. Function rcu\_per\_clock\_config

Function name	rcu_per_clock_config
Function prototype	void rcu_per_clock_config(uint32_t ck_per);
Function descriptions	configure the PER clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
ck_per	PER clock source selection
RCU_PERSRC_IRC64MDIV	select CK_IRC64MDIV as PER clock
RCU_PERSRC_LPIRC4M	select CK_LPIRC4M as PER clock
RCU_PERSRC_HXTAL	select CK_HXTAL as PER clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_IRC64MDIV as the PER clock source */
```

```
rcu_per_clock_config(RCU_PERSRC_IRC64MDIV);
```

### rcu\_usbhs\_pll1qpsc\_config

The description of rcu\_usbhs\_pll1qpsc\_config is shown as below:

Table 3-958. Function rcu\_usbhs\_pll1qpsc\_config

Function name	rcu_usbhs_pll1qpsc_config
Function prototype	void rcu_usbhs_pll1qpsc_config(usbhs_idx_enum usbhs_idx, uint32_t ck_usbhspsc);
Function descriptions	configure the PLL1Q prescaler
Precondition	-
The called functions	-
Input parameter{in}	
usbhs_idx	USBHS index, refer to <a href="#">Table 3-909. Enum usbhs_idx_enum</a>
Input parameter{in}	
ck_usbhspsc	USBHS clock prescaler from CK_PLL1Q
RCU_USBHSPSC_DIV $x(x=1,2...8)$	CK_PLL1Q / X
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the RCU_USBHSPSC_DIV as RCU_USBHSPSC_DIV1 */
rcu_usbhs_pll1qpsc_config(IDX_USBHS0, RCU_USBHSPSC_DIV1);
```

### rcu\_usb48m\_clock\_config

The description of rcu\_usb48m\_clock\_config is shown as below:

**Table 3-959. Function rcu\_usb48m\_clock\_config**

Function name	rcu_usb48m_clock_config
Function prototype	void rcu_usb48m_clock_config(usbhs_idx_enum usbhs_idx,uint32_t ck_usb48m);
Function descriptions	configure the USBHS48MSEL clock source selection
Precondition	-
The called functions	-
Input parameter{in}	
usbhs_idx	USBHS index, refer to <a href="#">Table 3-909. Enum usbhs_idx_enum</a>
Input parameter{in}	
ck_usb48m	USBHS48M clock source selection
RCU_USB48MSRC_PL L0R	select CK_PLL0R as USBHS48M clock
RCU_USB48MSRC_PL LUSBHS	select CK_PLLUSBHS as USBHS48M clock
RCU_USB48MSRC_PL L1Q	select CK_PLL1Q as USBHS48M clock
RCU_USB48MSRC_IR C48M	select CK_IRC48M as USBHS48M clock
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the CK_IRC48M as USBHS48M clock */
rcu_usb48m_clock_config(IDX_USBHS0, RCU_USB48MSRC_IRC48M);
```

### rcu\_usbhs\_clock\_config

The description of rcu\_usbhs\_clock\_config is shown as below:

**Table 3-960. Function rcu\_usbhs\_clock\_config**

Function name	rcu_usbhs_clock_config
Function prototype	void rcu_usbhs_clock_config(usbhs_idx_enum usbhs_idx,uint32_t ck_usbhs);



<b>Function descriptions</b>	configure the USBHSSEL clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usbhs_idx</b>	USBHS index, refer to <a href="#">Table 3-909. Enum usbhs_idx_enum</a>
<b>Input parameter{in}</b>	
<b>ck_usbhs</b>	USBHS clock source selection
<i>RCU_USBHSSEL_48M</i>	select 48M as USBHS clock
<i>RCU_USBHSSEL_60M</i>	select 60M as USBHS clock
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the 48M as USBHS clock */
```

```
rcu_usbhs_clock_config(IDX_USBHS0, RCU_USBHSSEL_48M);
```

### rcu\_usbhs\_clock\_selection\_enable

The description of rcu\_usbhs\_clock\_selection\_enable is shown as below:

**Table 3-961. Function rcu\_usbhs\_clock\_selection\_enable**

<b>Function name</b>	rcu_usbhs_clock_selection_enable
<b>Function prototype</b>	void rcu_usbhs_clock_selection_enable(usbhs_idx_enum usbhs_idx);
<b>Function descriptions</b>	enable the USBHS clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usbhs_idx</b>	USBHS index, refer to <a href="#">Table 3-909. Enum usbhs_idx_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable the USBHS0 clock source selection */
```

```
rcu_usbhs_clock_selection_enable(IDX_USBHS0);
```

### rcu\_usbhs\_clock\_selection\_disable

The description of rcu\_usbhs\_clock\_selection\_disable is shown as below:

**Table 3-962. Function rcu\_usbhs\_clock\_selection\_disable**

<b>Function name</b>	rcu_usbhs_clock_selection_disable
<b>Function prototype</b>	void rcu_usbhs_clock_selection_disable(usbhs_idx_enum usbhs_idx);
<b>Function descriptions</b>	disable the USBHS clock source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usbhs_idx</b>	USBHS index, refer to <a href="#">Table 3-909. Enum usbhs_idx_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the USBHS0 clock source selection */
rcu_usbhs_clock_selection_disable(IDX_USBHS0);
```

### rcu\_lxtal\_drive\_capability\_config

The description of rcu\_lxtal\_drive\_capability\_config is shown as below:

**Table 3-963. Function rcu\_lxtal\_drive\_capability\_config**

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

Example:

```
/* set the LXTAL lower driving capability */
rcu_lxtal_drive_capability_config(RCU_LXTAL_LOWDRI);
```

## rcu\_osci\_stab\_wait

The description of rcu\_osci\_stab\_wait is shown as below:

**Table 3-964. Function rcu\_osci\_stab\_wait**

<b>Function name</b>	rcu_osci_stab_wait
<b>Function prototype</b>	ErrStatus rcu_osci_stab_wait(rcu_osci_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 3-903. Enum rcu_osci_type_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	SUCCESS or ERROR

Example:

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

## rcu\_osci\_on

The description of rcu\_osci\_on is shown as below:

**Table 3-965. Function rcu\_osci\_on**

<b>Function name</b>	rcu_osci_on
<b>Function prototype</b>	void rcu_osci_on(rcu_osci_type_enum osci);
<b>Function descriptions</b>	turn on the oscillator
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>osci</b>	oscillator types, refer to <a href="#">Table 3-903. Enum rcu_osci_type_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## rcu\_osc\_off

The description of rcu\_osc\_off is shown as below:

**Table 3-966. Function rcu\_osc\_off**

<b>Function name</b>	rcu_osc_off
<b>Function prototype</b>	void rcu_osc_off(rcu_osc_type_enum osci);
<b>Function descriptions</b>	turn off the oscillator
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>osci</b>	oscillator types, refer to <a href="#">Table 3-903. Enum rcu_osc_type_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## rcu\_osc\_bypass\_mode\_enable

The description of rcu\_osc\_bypass\_mode\_enable is shown as below:

**Table 3-967. Function rcu\_osc\_bypass\_mode\_enable**

<b>Function name</b>	rcu_osc_bypass_mode_enable
<b>Function prototype</b>	void rcu_osc_bypass_mode_enable(rcu_osc_type_enum osci);
<b>Function descriptions</b>	enable the oscillator bypass mode
<b>Precondition</b>	HXTALEN or LXTALEN must be reset before it
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>osci</b>	oscillator types, refer to <a href="#">Table 3-903. Enum rcu_osc_type_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## rcu\_osc\_bypass\_mode\_disable

The description of rcu\_osc\_bypass\_mode\_disable is shown as below:

**Table 3-968. Function rcu\_osc\_bypass\_mode\_disable**

<b>Function name</b>	rcu_osc_bypass_mode_disable
<b>Function prototype</b>	void rcu_osc_bypass_mode_disable(rcu_osc_type_enum osci);
<b>Function descriptions</b>	disable the oscillator bypass mode
<b>Precondition</b>	HXTALEN or LXTALEN must be reset before it
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>osci</b>	oscillator types, refer to <a href="#">Table 3-903. Enum rcu_osc_type_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable the high speed crystal oscillator bypass mode */
```

```
rcu_osc_bypass_mode_disable(RCU_HXTAL);
```

### rcu\_irc64m\_adjust\_value\_set

The description of rcu\_irc64m\_adjust\_value\_set is shown as below:

**Table 3-969. Function rcu\_irc64m\_adjust\_value\_set**

<b>Function name</b>	rcu_irc64m_adjust_value_set
<b>Function prototype</b>	void rcu_irc64m_adjust_value_set(uint32_t irc64m_adjval);
<b>Function descriptions</b>	set the IRC64M adjust value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>irc64m_adjval</b>	IRC64M adjust value, must be between 0 and 0x7F
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the IRC64M adjust value */
```

```
rcu_irc8m_adjust_value_set(0x20);
```

### rcu\_lpirc4m\_adjust\_value\_set

The description of rcu\_lpirc4m\_adjust\_value\_set is shown as below:

**Table 3-970. Function rcu\_lpirc4m\_adjust\_value\_set**

<b>Function name</b>	rcu_lpirc4m_adjust_value_set
----------------------	------------------------------

<b>Function prototype</b>	void rcu_lpirc4m_adjust_value_set(uint32_t lpirc4m_adjval);
<b>Function descriptions</b>	set the LPIRC4M adjust value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lpirc4m_adjval</b>	LPIRC4M adjust value, must be between 0 and 0x3F
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the LPIRC4M adjust value */
rcu_lpirc4m_adjust_value_set(0x10);
```

### rcu\_hxtal\_clock\_monitor\_enable

The description of rcu\_hxtal\_clock\_monitor\_enable is shown as below:

**Table 3-971. 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 3-972. 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\_lxtal\_clock\_monitor\_enable

The description of rcu\_lxtal\_clock\_monitor\_enable is shown as below:

**Table 3-973. Function rcu\_lxtal\_clock\_monitor\_enable**

<b>Function name</b>	rcu_lxtal_clock_monitor_enable
<b>Function prototype</b>	void rcu_lxtal_clock_monitor_enable(void);
<b>Function descriptions</b>	enable the LXTAL 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 LXTAL clock monitor */
```

```
rcu_lxtal_clock_monitor_enable();
```

### rcu\_lxtal\_clock\_monitor\_disable

The description of rcu\_lxtal\_clock\_monitor\_disable is shown as below:

**Table 3-974. Function rcu\_lxtal\_clock\_monitor\_disable**

<b>Function name</b>	rcu_lxtal_clock_monitor_disable
<b>Function prototype</b>	void rcu_lxtal_clock_monitor_disable(void);
<b>Function descriptions</b>	disable the LXTAL clock monitor
<b>Precondition</b>	-
<b>The called functions</b>	-

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

Example:

```
/* disable the LXTAL clock monitor */
```

```
rcu_lxtal_clock_monitor_disable();
```

### rcu\_clock\_freq\_get

The description of rcu\_clock\_freq\_get is shown as below:

**Table 3-975. Function rcu\_clock\_freq\_get**

Function name	rcu_clock_freq_get
Function prototype	uint32_t rcu_clock_freq_get(rcu_clock_freq_enum clock);
Function descriptions	get the system clock, bus clock and peripheral clock frequency
Precondition	-
The called functions	-
Input parameter{in}	
clock	the clock frequency which to get, refer to <a href="#">Table 3-904. Enum rcu_clock_freq_enum</a>
Output parameter{out}	
-	-
Return value	
uint32_t	clock frequency of system, AHB, APB1, APB2, APB3, APB4, PLL or USRAT

Example:

```
uint32_t temp_freq;
```

```
/* get the system clock frequency */
```

```
temp_freq = rcu_clock_freq_get(CK_SYS);
```

### rcu\_flag\_get

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

**Table 3-976. Function rcu\_flag\_get**

Function name	rcu_flag_get
Function prototype	FlagStatus rcu_flag_get(rcu_flag_enum flag);
Function descriptions	get the clock stabilization and peripheral reset flags
Precondition	-



The called functions	-
Input parameter{in}	
flag	the clock stabilization and peripheral reset flags, refer to <a href="#">Table 3-899. Enum rcu_flag_enum</a>
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

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

### rcu\_all\_reset\_flag\_clear

The description of rcu\_all\_reset\_flag\_clear is shown as below:

**Table 3-977. Function rcu\_all\_reset\_flag\_clear**

Function name	rcu_all_reset_flag_clear
Function prototype	void rcu_all_reset_flag_clear(void);
Function descriptions	clear all the reset flag
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### rcu\_interrupt\_enable

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

**Table 3-978. Function rcu\_interrupt\_enable**

Function name	rcu_interrupt_enable
Function prototype	void rcu_interrupt_enable(rcu_int_enum interrupt);
Function descriptions	enable the stabilization interrupt
Precondition	-

The called functions	-
Input parameter{in}	
interrupt	clock stabilization interrupt, refer to <a href="#">Table 3-902. Enum rcu_int_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### rcu\_interrupt\_disable

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

**Table 3-979. Function rcu\_interrupt\_disable**

Function name	rcu_interrupt_disable
Function prototype	void rcu_interrupt_disable(rcu_int_enum interrupt);
Function descriptions	disable the stabilization interrupt
Precondition	-
The called functions	-
Input parameter{in}	
interrupt	clock stabilization interrupt, refer to <a href="#">Table 3-902. Enum rcu_int_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

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

### rcu\_interrupt\_flag\_get

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

**Table 3-980. Function rcu\_interrupt\_flag\_get**

Function name	rcu_interrupt_flag_get
Function prototype	FlagStatus rcu_interrupt_flag_get(rcu_int_flag_enum int_flag);
Function descriptions	get the clock stabilization interrupt and ckm flags
Precondition	-
The called functions	-
Input parameter{in}	

<b>int_flag</b>	interrupt and ckm flags, refer to <a href="#">Table 3-900. Enum rcu_int_flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

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

### rcu\_interrupt\_flag\_clear

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

**Table 3-981. 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 3-901. Enum rcu_int_flag_clear_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

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

## 3.29. RTC

The Real-time Clock (RTC) is usually used as a clock-calendar. The ones in the Backup Domain consist of a 32-bit up-counter, an alarm, a prescaler, a divider and the RTC clock configuration register. The RTC registers are listed in chapter [3.29.1](#), the FWDGT firmware functions are introduced in chapter [3.29.2](#).

### 3.29.1. Descriptions of Peripheral registers

RTC registers are listed in the table shown as below:

**Table 3-982. RTC Registers**

Registers	Descriptions
RTC_TIME	RTC time of day register
RTC_DATE	RTC date register
RTC_CTL	RTC control register
RTC_STAT	RTC status register
RTC_PSC	RTC time prescaler register
RTC_WUT	RTC wakeup timer register
RTC_ALRM0TD	RTC alarm 0 time and date register
RTC_ALRM1TD	RTC alarm 1 time and date register
RTC_WPK	RTC write protection key register
RTC_SS	RTC sub second register
RTC_SHIFTCTL	RTC shift function control register
RTC_TTS	RTC time of timestamp register
RTC_DTS	RTC date of timestamp register
RTC_SSTS	RTC sub second of timestamp register
RTC_HRFC	RTC high resolution frequency compensation register
RTC_TAMP	RTC tamper register
RTC_ALRM0SS	RTC alarm 0 sub second register
RTC_ALRM1SS	RTC alarm 1 sub second register
RTC_CFG	RTC configure register
RTC_BKP0	RTC backup 0 register
RTC_BKP1	RTC backup 1 register
RTC_BKP2	RTC backup 2 register
RTC_BKP3	RTC backup 3 register
RTC_BKP4	RTC backup 4 register
RTC_BKP5	RTC backup 5 register
RTC_BKP6	RTC backup 6 register
RTC_BKP7	RTC backup 7 register
RTC_BKP8	RTC backup 8 register
RTC_BKP9	RTC backup 9 register
RTC_BKP10	RTC backup 10 register
RTC_BKP11	RTC backup 11 register
RTC_BKP12	RTC backup 12 register
RTC_BKP13	RTC backup 13 register
RTC_BKP14	RTC backup 14 register
RTC_BKP15	RTC backup 15 register
RTC_BKP16	RTC backup 16 register
RTC_BKP17	RTC backup 17 register
RTC_BKP18	RTC backup 18 register
RTC_BKP19	RTC backup 19 register
RTC_BKP20	RTC backup 20 register
RTC_BKP21	RTC backup 21 register

Registers	Descriptions
RTC_BKP22	RTC backup 22 register
RTC_BKP23	RTC backup 23 register
RTC_BKP24	RTC backup 24 register
RTC_BKP25	RTC backup 25 register
RTC_BKP26	RTC backup 26 register
RTC_BKP27	RTC backup 27 register
RTC_BKP28	RTC backup 28 register
RTC_BKP29	RTC backup 29 register
RTC_BKP30	RTC backup 30 register
RTC_BKP31	RTC backup 31 register

### 3.29.2. Descriptions of Peripheral functions

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

**Table 3-983. RTC firmware function**

Function name	Function description
rtc_deinit	reset most of the RTC registers
rtc_init	initialize RTC registers
rtc_init_mode_enter	enter RTC init mode
rtc_init_mode_exit	exit RTC init mode
rtc_register_sync_wait	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
rtc_current_time_get	get current time and date
rtc_subsecond_get	get current subsecond value
rtc_alarm_config	configure RTC alarm
rtc_alarm_subsecond_config	configure subsecond of RTC alarm
rtc_alarm_get	get RTC alarm
rtc_alarm_subsecond_get	get RTC alarm subsecond
rtc_alarm_enable	enable RTC alarm
rtc_alarm_disable	disable RTC alarm
rtc_timestamp_enable	enable RTC time-stamp
rtc_timestamp_disable	disable RTC time-stamp
rtc_timestamp_get	get RTC timestamp time and date
rtc_timestamp_internalevent_config	configure RTC time-stamp internal event
rtc_timestamp_subsecond_get	get RTC time-stamp subsecond
rtc_tamper_enable	enable RTC tamper
rtc_tamper_disable	disable RTC tamper
rtc_output_pin_select	select the RTC output pin
rtc_alarm_output_config	configure RTC alarm output source
rtc_calibration_output_config	configure RTC calibration output source
rtc_hour_adjust	adjust the daylight saving time by adding or subtracting one

Function name	Function description
	hour from the current time
rtc_second_adjust	adjust RTC second or subsecond value of current time
rtc_bypass_shadow_enable	enable RTC bypass shadow registers function
rtc_bypass_shadow_disable	disable RTC bypass shadow registers function
rtc_refclock_detection_enable	enable RTC reference clock detection function
rtc_refclock_detection_disable	disable RTC reference clock detection function
rtc_wakeup_enable	enable RTC wakeup timer
rtc_wakeup_disable	disable RTC wakeup timer
rtc_wakeup_clock_set	set auto wakeup timer clock
rtc_wakeup_timer_set	set auto wakeup timer value
rtc_wakeup_timer_get	get auto wakeup timer value
rtc_smooth_calibration_config	configure RTC smooth calibration
rtc_interrupt_enable	enable specified RTC interrupt
rtc_interrupt_disable	disable specified RTC interrupt
rtc_flag_get	check specified flag
rtc_flag_clear	clear specified flag

### Structure rtc\_parameter\_struct

**Table 3-984. Structure rtc\_parameter\_struct**

Member name	Function description
year	RTC year value: 0x0 - 0x99(BCD format)
month	RTC month value (BCD format)
date	RTC date value: 0x1 - 0x31(BCD format)
day_of_week	RTC weekday value(BCD format)
hour	RTC hour value: 0x1 - 0x12(BCD format) or 0x0 - 0x23(BCD format)
minute	RTC minute value: 0x0 - 0x59(BCD format)
second	RTC second value: 0x0 - 0x59(BCD format)
factor_asyn	RTC asynchronous prescaler value: 0x0 - 0x7F
factor_syn	RTC synchronous prescaler value: 0x0 - 0x7FFF
am_pm	RTC AM/PM value
display_format	RTC time notation

### Structure rtc\_alarm\_struct

**Table 3-985. Structure rtc\_alarm\_struct**

Member name	Function description
alarm_mask	RTC alarm mask
weekday_or_date	specify RTC alarm is on date or weekday
alarm_day	RTC alarm date or weekday value(BCD format)
alarm_hour	RTC alarm hour value: 0x1 - 0x12(BCD format) or 0x0 - 0x23(BCD format)
alarm_minute	RTC alarm minute value: 0x0 - 0x59(BCD format)
alarm_second	RTC alarm second value: 0x0 - 0x59(BCD format)

am_pm	RTC alarm AM/PM value
-------	-----------------------

### Structure rtc\_timestamp\_struct

**Table 3-986. Structure rtc\_timestamp\_struct**

Member name	Function description
timestamp_month	RTC time-stamp month value(BCD format)
timestamp_date	RTC time-stamp date value: 0x1 - 0x31(BCD format)
timestamp_day	RTC time-stamp weekday value(BCD format)
timestamp_hour	RTC time-stamp hour value(BCD format): 0x1 - 0x12(BCD format) or 0x0 - 0x23(BCD format)
timestamp_minute	RTC time-stamp minute value: 0x0 - 0x59(BCD format)
timestamp_second	RTC time-stamp second value: 0x0 - 0x59(BCD format)
am_pm	RTC time-stamp AM/PM value

### Structure rtc\_tamper\_struct

**Table 3-987. Structure rtc\_tamper\_struct**

Member name	Function description
tamper_source	RTC tamper source
tamper_trigger	RTC tamper trigger
tamper_filter	RTC tamper consecutive samples needed during a voltage level detection
tamper_sample_frequency	RTC tamper sampling frequency during a voltage level detection
tamper_precharge_enable	RTC tamper precharge feature during a voltage level detection
tamper_precharge_time	RTC tamper precharge duration if precharge feature is enabled
tamper_with_timestamp	RTC tamper time-stamp feature

### rtc\_deinit

The description of rtc\_deinit is shown as below:

**Table 3-988. Function rtc\_deinit**

<b>Function name</b>	rtc_deinit
<b>Function prototype</b>	ErrStatus rtc_deinit(void);
<b>Function descriptions</b>	reset most of the RTC registers
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable/ rcu_periph_reset_disable -
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-

Return value	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* reset most of the RTC registers*/
```

```
ErrStatus error_status = rtc_deinit();
```

## rtc\_init

The description of rtc\_init is shown as below:

**Table 3-989. Function rtc\_init**

<b>Function name</b>	rtc_init
<b>Function prototype</b>	ErrStatus rtc_init(rtc_parameter_struct* rtc_initpara_struct);
<b>Function descriptions</b>	initialize RTC registers
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_initpara_struct</b>	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure <a href="#">Table 3-984. Structure rtc_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* initialize RTC registers */
```

```
rtc_parameter_struct rtc_initpara;
```

```
rtc_interrupt_disable(RTC_INT_SECOND);
```

```
rtc_initpara.factor_asyn = prescaler_a;
```

```
rtc_initpara.factor_syn = prescaler_s;
```

```
rtc_initpara.year = 0x16;
```

```
rtc_initpara.day_of_week = RTC_SATURDAY;
```

```
rtc_initpara.month = RTC_APR;
```

```
rtc_initpara.date = 0x30;
```

```
rtc_initpara.display_format = RTC_24HOUR;
```

```
rtc_initpara.am_pm = RTC_AM;
```



```
rtc_init(&rtc_initpara);
```

### rtc\_init\_mode\_enter

The description of rtc\_init\_mode\_enter is shown as below:

**Table 3-990. Function rtc\_init\_mode\_enter**

<b>Function name</b>	rtc_init_mode_enter
<b>Function prototype</b>	ErrStatus rtc_init_mode_enter(void);
<b>Function descriptions</b>	enter RTC init mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/*enter RTC init mode*/
```

```
ErrStatus error_status = rtc_init_mode_enter();
```

### rtc\_init\_mode\_exit

The description of rtc\_init\_mode\_exit is shown as below:

**Table 3-991. Function rtc\_init\_mode\_exit**

<b>Function name</b>	rtc_init_mode_exit
<b>Function prototype</b>	void rtc_init_mode_exit(void);
<b>Function descriptions</b>	exit RTC init mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*exit RTC init mode*/
```

```
rtc_init_mode_exit();
```

## rtc\_register\_sync\_wait

The description of rtc\_register\_sync\_wait is shown as below:

**Table 3-992. Function rtc\_register\_sync\_wait**

<b>Function name</b>	rtc_register_sync_wait
<b>Function prototype</b>	ErrStatus rtc_register_sync_wait(void);
<b>Function descriptions</b>	wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the shadow registers are updated
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/*wait until RTC_TIME and RTC_DATE registers are synchronized with APB clock, and the
shadow registers are updated*/
```

```
ErrStatus error_status = rtc_register_sync_wait();
```

## rtc\_current\_time\_get

The description of rtc\_current\_time\_get is shown as below:

**Table 3-993. Function rtc\_current\_time\_get**

<b>Function name</b>	rtc_current_time_get
<b>Function prototype</b>	void rtc_current_time_get(rtc_parameter_struct* rtc_initpara_struct);
<b>Function descriptions</b>	get current time and date
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>rtc_initpara_struct</b>	pointer to a rtc_parameter_struct structure which contains parameters for initialization of the rtc peripheral, the structure members can refer to members of the structure <a href="#">Table 3-984. Structure rtc_parameter_struct</a>
<b>Return value</b>	
-	-

Example:

```
/*get current time and date*/
```

```
rtc_parameter_struct rtc_initpara_struct;

rtc_current_time_get(&rtc_initpara_struct);
```

### rtc\_subsecond\_get

The description of rtc\_subsecond\_get is shown as below:

**Table 3-994. Function rtc\_subsecond\_get**

<b>Function name</b>	rtc_subsecond_get
<b>Function prototype</b>	uint32_t rtc_subsecond_get(void);
<b>Function descriptions</b>	get current subsecond value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	current subsecond value(0x00-0xFFFF)

Example:

```
/*get current subsecond value*/

uint32_t sub_second = rtc_subsecond_get();
```

### rtc\_alarm\_config

The description of rtc\_alarm\_config is shown as below:

**Table 3-995. Function rtc\_alarm\_config**

<b>Function name</b>	rtc_alarm_config
<b>Function prototype</b>	void rtc_alarm_config(uint8_t rtc_alarm, rtc_alarm_struct *rtc_alarm_time);
<b>Function descriptions</b>	configure RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
rtc_alarm	RTC_ALARM0 or RTC_ALARM1
<b>Input parameter{in}</b>	
rtc_alarm_time	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure <a href="#">Table 3-985. Structure rtc_alarm_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*rtc_alarm_config*/

rtc_alarm_struct rtc_alarm_time;

rtc_alarm_config(RTC_ALARM0,&rtc_alarm_time);
```

### rtc\_alarm\_subsecond\_config

The description of rtc\_alarm\_subsecond\_config is shown as below:

**Table 3-996. Function rtc\_alarm\_subsecond\_config**

Function name	rtc_alarm_subsecond_config
Function prototype	void rtc_alarm_subsecond_config(uint8_t rtc_alarm, uint32_t mask_subsecond, uint32_t subsecond)
Function descriptions	configure subsecond of RTC alarm
Precondition	-
The called functions	-
Input parameter{in}	
rtc_alarm	RTC_ALARM0 or RTC_ALARM1
Input parameter{in}	
mask_subsecond	alarm subsecond mask
RTC_MASKSSC_0_14	mask alarm subsecond configuration
RTC_MASKSSC_1_14	mask RTC_ALARM0SS_SSC[14:1], and RTC_ALARM0SS_SSC[0] is to be compared
RTC_MASKSSC_2_14	mask RTC_ALARM0SS_SSC[14:2], and RTC_ALARM0SS_SSC[1:0] is to be compared
RTC_MASKSSC_3_14	mask RTC_ALARM0SS_SSC[14:3], and RTC_ALARM0SS_SSC[2:0] is to be compared
RTC_MASKSSC_4_14	mask RTC_ALARM0SS_SSC[14:4], and RTC_ALARM0SS_SSC[3:0] is to be compared
RTC_MASKSSC_5_14	mask RTC_ALARM0SS_SSC[14:5], and RTC_ALARM0SS_SSC[4:0] is to be compared
RTC_MASKSSC_6_14	mask RTC_ALARM0SS_SSC[14:6], and RTC_ALARM0SS_SSC[5:0] is to be compared
RTC_MASKSSC_7_14	mask RTC_ALARM0SS_SSC[14:7], and RTC_ALARM0SS_SSC[6:0] is to be compared
RTC_MASKSSC_8_14	mask RTC_ALARM0SS_SSC[14:8], and RTC_ALARM0SS_SSC[7:0] is to be compared
RTC_MASKSSC_9_14	mask RTC_ALARM0SS_SSC[14:9], and RTC_ALARM0SS_SSC[8:0] is to be compared
RTC_MASKSSC_10_14	mask RTC_ALARM0SS_SSC[14:10], and RTC_ALARM0SS_SSC[9:0] is to be compared
RTC_MASKSSC_11_14	mask RTC_ALARM0SS_SSC[14:11], and RTC_ALARM0SS_SSC[10:0] is to be compared

<i>RTC_MASKSSC_12_1</i> 4	mask RTC_ALARM0SS_SSC[14:12], and RTC_ALARM0SS_SSC[11:0] is to be compared
<i>RTC_MASKSSC_13_1</i> 4	mask RTC_ALARM0SS_SSC[14:13], and RTC_ALARM0SS_SSC[12:0] is to be compared
<i>RTC_MASKSSC_14</i>	mask RTC_ALARM0SS_SSC[14], and RTC_ALARM0SS_SSC[13:0] is to be compared
<i>RTC_MASKSSC_NONE</i>	mask none, and RTC_ALARM0SS_SSC[14:0] is to be compared
<b>Input parameter{in}</b>	
<b>subsecond</b>	alarm subsecond value(0x000 - 0x7FFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*configure subsecond of RTC alarm*/
```

```
rtc_subsecond_config(RTC_ALARM0,RTC_MASKSSC_9_14, 0x7FFF);
```

### rtc\_alarm\_enable

The description of rtc\_alarm\_enable is shown as below:

**Table 3-997. Function rtc\_alarm\_enable**

<b>Function name</b>	rtc_alarm_enable
<b>Function prototype</b>	void rtc_alarm_enable(uint8_t rtc_alarm);
<b>Function descriptions</b>	enable RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*enable RTC alarm*/
```

```
rtc_alarm_enable(RTC_ALARM0);
```

### rtc\_alarm\_disable

The description of rtc\_alarm\_disable is shown as below:

**Table 3-998. Function rtc\_alarm\_disable**

<b>Function name</b>	rtc_alarm_disable
<b>Function prototype</b>	ErrStatus rtc_alarm_disable(uint8_t rtc_alarm);
<b>Function descriptions</b>	disable RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/*disable RTC alarm*/
```

```
ErrStatus error_status = rtc_alarm_disable(RTC_ALARM0);
```

### rtc\_alarm\_get

The description of rtc\_alarm\_get is shown as below:

**Table 3-999. Function rtc\_alarm\_get**

<b>Function name</b>	rtc_alarm_get
<b>Function prototype</b>	void rtc_alarm_get(uint8_t rtc_alarm, rtc_alarm_struct *rtc_alarm_time);
<b>Function descriptions</b>	get RTC alarm
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Output parameter{out}</b>	
<b>rtc_alarm_time</b>	pointer to a rtc_alarm_struct structure which contains parameters for RTC alarm configuration, the structure members can refer to members of the structure <a href="#">Table 3-985. Structure rtc_alarm_struct</a>
<b>Return value</b>	
-	-

Example:

```
/* get RTC alarm */
```

```
rtc_alarm_struct rtc_alarm_time;
```

```
rtc_alarm_get(RTC_ALARM0,&rtc_alarm_time);
```

## rtc\_alarm\_subsecond\_get

The description of rtc\_alarm\_subsecond\_get is shown as below:

**Table 3-1000. Function rtc\_alarm\_subsecond\_get**

<b>Function name</b>	rtc_alarm_subsecond_get
<b>Function prototype</b>	uint32_t rtc_alarm_subsecond_get(uint8_t rtc_alarm);
<b>Function descriptions</b>	get RTC alarm subsecond
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_alarm</b>	RTC_ALARM0 or RTC_ALARM1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	RTC alarm subsecond value(0x0-0x3FFF)

Example:

```
/*get RTC alarm subsecond*/
```

```
uint32_t subsecond = rtc_alarm_subsecond_get(RTC_ALARM0);
```

## rtc\_timestamp\_enable

The description of rtc\_timestamp\_enable is shown as below:

**Table 3-1001. Function rtc\_timestamp\_enable**

<b>Function name</b>	rtc_timestamp_enable
<b>Function prototype</b>	void rtc_timestamp_enable(uint32_t edge);
<b>Function descriptions</b>	enable RTC time-stamp
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>edge</b>	specify which edge to detect of time-stamp
<i>RTC_TIMESTAMP_RISING_EDGE</i>	rising edge is valid event edge for timestamp event
<i>RTC_TIMESTAMP_FALLING_EDGE</i>	falling edge is valid event edge for timestamp event
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*enable RTC time-stamp*/
```

```
rtc_timestamp_enable(RTC_TIMESTAMP_RISING_EDGE);
```

## rtc\_timestamp\_disable

The description of `rtc_timestamp_disable` is shown as below:

**Table 3-1002. Function `rtc_timestamp_disable`**

<b>Function name</b>	<code>rtc_timestamp_disable</code>
<b>Function prototype</b>	<code>void rtc_timestamp_disable(void);</code>
<b>Function descriptions</b>	disable RTC time-stamp
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*disable RTC time-stamp*/
```

```
rtc_timestamp_disable();
```

## rtc\_timestamp\_internalevent\_config

The description of `rtc_timestamp_internalevent_config` is shown as below:

**Table 3-1003. Function `rtc_timestamp_internalevent_config`**

<b>Function name</b>	<code>rtc_timestamp_internalevent_config</code>
<b>Function prototype</b>	<code>void rtc_timestamp_internalevent_config(uint32_t mode)</code>
<b>Function descriptions</b>	configure RTC time-stamp internal event
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mode</b>	specify which internal or external event to be detected
<i>RTC_ITSEN_DISABLE</i>	disable RTC time-stamp internal event
<i>RTC_ITSEN_ENABLE</i>	enable RTC time-stamp internal event
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable RTC time-stamp internal event */
```



```
rtc_timestamp_internalevent_config(RTC_ITSEN_DISABLE);
```

## rtc\_timestamp\_get

The description of rtc\_timestamp\_get is shown as below:

**Table 3-1004. Function rtc\_timestamp\_get**

<b>Function name</b>	rtc_timestamp_get
<b>Function prototype</b>	void rtc_timestamp_get(rtc_timestamp_struct* rtc_timestamp);
<b>Function descriptions</b>	get RTC timestamp time and date
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
rtc_timestamp	Pointer to a rtc_timestamp_struct structure which contains parameters for RTC time-stamp configuration, the structure members can refer to members of the structure <a href="#">Table 3-987. Structure rtc_tamper_struct</a>
<b>Return value</b>	
-	-

Example:

```
/* get RTC timestamp time and date */
rtc_timestamp_struct rtc_timestamp;
rtc_timestamp_get(& rtc_timestamp);
```

## rtc\_timestamp\_subsecond\_get

The description of rtc\_timestamp\_subsecond\_get is shown as below:

**Table 3-1005. Function rtc\_timestamp\_subsecond\_get**

<b>Function name</b>	rtc_timestamp_subsecond_get
<b>Function prototype</b>	uint32_t rtc_timestamp_subsecond_get(void);
<b>Function descriptions</b>	get RTC time-stamp subsecond
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	RTC time-stamp subsecond value

Example:

```
/* get RTC time-stamp subsecond */
```

```
uint32_t subsecond = rtc_timestamp_subsecond_get();
```

## rtc\_tamper\_enable

The description of rtc\_tamper\_enable is shown as below:

**Table 3-1006. Function rtc\_tamper\_enable**

<b>Function name</b>	rtc_tamper_enable
<b>Function prototype</b>	void rtc_tamper_enable(rtc_tamper_struct* rtc_tamper);
<b>Function descriptions</b>	enable RTC tamper
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>rtc_tamper</b>	pointer to a rtc_tamper_struct structure which contains parameters for RTC tamper configuration, the structure members can refer to members of the structure <a href="#">Table 3-987. Structure rtc_tamper_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable RTC tamper */
```

```
rtc_tamper_struct rtc_tamper
```

```
rtc_tamper_enable(& rtc_tamper);
```

## rtc\_tamper\_disable

The description of rtc\_tamper\_disable is shown as below:

**Table 3-1007. Function rtc\_tamper\_disable**

<b>Function name</b>	rtc_tamper_disable
<b>Function prototype</b>	void rtc_tamper_disable(uint32_t source);
<b>Function descriptions</b>	disable RTC tamper
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>source</b>	specify which tamper source to be disabled
<i>RTC_TAMPER0</i>	RTC tamper0
<i>RTC_TAMPER1</i>	RTC tamper1
<i>RTC_TAMPER2</i>	RTC tamper2
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* disable RTC tamper0 */
rtc_tamper_disable(RTC_TAMPER0);
```

### rtc\_output\_pin\_select

The description of rtc\_output\_pin\_select is shown as below:

**Table 3-1008. Function rtc\_output\_pin\_select**

Function name	rtc_output_pin_select
Function prototype	void rtc_output_pin_select(uint32_t outputpin);
Function descriptions	select the RTC output pin
Precondition	-
The called functions	-
Input parameter{in}	
outputpin	specify the rtc output pin is PC13 or PB2
<i>RTC_OUT_PC13</i>	the rtc output pin is PC13
<i>RTC_OUT_PB2</i>	the rtc output pin is PB2
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* specify the rtc output pin is PC13 */
rtc_output_pin_select(RTC_OUT_PC13);
```

### rtc\_alarm\_output\_config

The description of rtc\_alarm\_output\_config is shown as below:

**Table 3-1009. Function rtc\_alarm\_output\_config**

Function name	rtc_alarm_output_config
Function prototype	void rtc_alarm_output_config(uint32_t source, uint32_t mode);
Function descriptions	configure rtc alarm output source
Precondition	-
The called functions	-
Input parameter{in}	
source	specify signal to output
<i>RTC_ALARM0_HIGH</i>	when the alarm0 flag is set, the output pin is high
<i>RTC_ALARM0_LOW</i>	when the alarm0 flag is set, the output pin is low

<i>RTC_ALARM1_HIGH</i>	when the alarm1 flag is set, the output pin is high
<i>RTC_ALARM1_LOW</i>	when the alarm1 flag is set, the output pin is low
<i>RTC_WAKEUP_HIGH</i>	when the wakeup flag is set, the output pin is high
<i>RTC_WAKEUP_LOW</i>	when the wakeup flag is set, the output pin is low
<b>Input parameter{in}</b>	
<b>mode</b>	specify the output pin mode when output alarm signal or auto wakeup signal
<i>RTC_ALARM_OUTPU</i> <i>T_OD</i>	open drain mode
<i>RTC_ALARM_OUTPU</i> <i>T_PP</i>	push pull mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure rtc alarm0 output source */
rtc_alarm_output_config(RTC_ALARM0_LOW, RTC_ALARM_OUTPUT_PP);
```

### rtc\_calibration\_output\_config

The description of `rtc_calibration_output_config` is shown as below:

**Table 3-1010. Function `rtc_calibration_output_config`**

<b>Function name</b>	<code>rtc_calibration_output_config</code>
<b>Function prototype</b>	<code>void rtc_calibration_output_config(uint32_t source);</code>
<b>Function descriptions</b>	configure rtc calibration output source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>source</b>	specify signal to output
<i>RTC_CALIBRATION_5</i> <i>12HZ</i>	when the LSE frequency is 32768Hz and the <code>RTC_PSC</code> is the default value, output 512Hz signal
<i>RTC_CALIBRATION_1</i> <i>HZ</i>	when the LSE frequency is 32768Hz and the <code>RTC_PSC</code> is the default value, output 1Hz signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* when the LSE frequency is 32768Hz and the RTC_PSC
is the default value, output 1Hz signal */
```

```
rtc_calibration_output_config(RTC_CALIBRATION_1HZ);
```

## rtc\_hour\_adjust

The description of rtc\_hour\_adjust is shown as below:

**Table 3-1011. Function rtc\_hour\_adjust**

<b>Function name</b>	rtc_hour_adjust
<b>Function prototype</b>	void rtc_hour_adjust(uint32_t operation);
<b>Function descriptions</b>	adjust the daylight saving time by adding or subtracting one hour from the current time
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>operation</b>	hour adjustment operation
<i>RTC_CTL_A1H</i>	add one hour
<i>RTC_CTL_S1H</i>	subtract one hour
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* adjust the daylight saving time by adding one hour from the current time */
```

```
rtc_hour_adjust(RTC_CTL_A1H);
```

## rtc\_second\_adjust

The description of rtc\_second\_adjust is shown as below:

**Table 3-1012. Function rtc\_second\_adjust**

<b>Function name</b>	rtc_second_adjust
<b>Function prototype</b>	ErrStatus rtc_second_adjust(uint32_t add, uint32_t minus);
<b>Function descriptions</b>	adjust RTC second or subsecond value of current time
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>add</b>	add 1s to current time or not
<i>RTC_SHIFT_ADD1S_R ESET</i>	no effect
<i>RTC_SHIFT_ADD1S_S ET</i>	add 1s to current time
<b>Input parameter{in}</b>	
<b>minus</b>	number of subsecond to minus from current time(0x0 - 0x7FFF)
<b>Output parameter{out}</b>	

-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* adjust RTC second or subsecond value of current time */
```

```
ErrStatus error_status = rtc_second_adjust(RTC_SHIFT_ADD1S_SET, 0);
```

### rtc\_bypass\_shadow\_enable

The description of rtc\_bypass\_shadow\_enable is shown as below:

**Table 3-1013. Function rtc\_bypass\_shadow\_enable**

<b>Function name</b>	rtc_bypass_shadow_enable
<b>Function prototype</b>	void rtc_bypass_shadow_enable(void);
<b>Function descriptions</b>	enable RTC bypass shadow registers function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable RTC bypass shadow registers function*/
```

```
rtc_bypass_shadow_enable();
```

### rtc\_bypass\_shadow\_disable

The description of rtc\_bypass\_shadow\_disable is shown as below:

**Table 3-1014. Function rtc\_bypass\_shadow\_disable**

<b>Function name</b>	rtc_bypass_shadow_disable
<b>Function prototype</b>	void rtc_bypass_shadow_disable(void);
<b>Function descriptions</b>	disable RTC bypass shadow registers function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable RTC bypass shadow registers function */
```

```
rtc_bypass_shadow_disable();
```

### rtc\_refclock\_detection\_enable

The description of rtc\_refclock\_detection\_enable shown as below:

**Table 3-1015. Function rtc\_refclock\_detection\_enable**

<b>Function name</b>	rtc_refclock_detection_enable
<b>Function prototype</b>	ErrStatus rtc_refclock_detection_enable(void);
<b>Function descriptions</b>	enable RTC reference clock detection function
<b>Precondition</b>	-
<b>The called functions</b>	rtc_init_mode_enter/rtc_init_mode_exit
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* enable RTC reference clock detection function*/
```

```
ErrStatus error_status = rtc_refclock_detection_enable();
```

### rtc\_refclock\_detection\_disable

The description of rtc\_refclock\_detection\_disable shown as below:

**Table 3-1016. Function rtc\_refclock\_detection\_disable**

<b>Function name</b>	rtc_refclock_detection_disable
<b>Function prototype</b>	ErrStatus rtc_refclock_detection_disable(void);
<b>Function descriptions</b>	disable RTC reference clock detection function
<b>Precondition</b>	-
<b>The called functions</b>	rtc_init_mode_enter/rtc_init_mode_exit
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* disableRTC reference clock detection function*/
```

```
ErrStatus error_status = rtc_refclock_detection_disable();
```

### rtc\_wakeup\_enable

The description of rtc\_refclock\_detection\_disable is shown as below:

**Table 3-1017. Function rtc\_wakeup\_enable**

<b>Function name</b>	rtc_wakeup_enable
<b>Function prototype</b>	void rtc_wakeup_enable(void);
<b>Function descriptions</b>	enable RTC auto wakeup function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable RTC auto wakeup function */
```

```
rtc_wakeup_enable();
```

### rtc\_wakeup\_disable

The description of rtc\_wakeup\_disable is shown as below:

**Table 3-1018. Function rtc\_wakeup\_disable**

<b>Function name</b>	rtc_wakeup_disable
<b>Function prototype</b>	ErrStatus rtc_wakeup_disable(void);
<b>Function descriptions</b>	disable RTC auto wakeup function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* disable RTC auto wakeup function */
```

```
ErrStatus error_status = rtc_wakeup_disable();
```



## rtc\_wakeup\_clock\_set

The description of rtc\_wakeup\_clock\_set is shown as below:

**Table 3-1019. Function rtc\_wakeup\_clock\_set**

<b>Function name</b>	rtc_wakeup_clock_set
<b>Function prototype</b>	ErrStatus rtc_wakeup_clock_set(uint8_t wakeup_clock);
<b>Function descriptions</b>	set RTC auto wakeup timer clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wakeup_clock</b>	wakeup timer clock is RTC clock divided factor
WAKEUP_RTCK_DIV 16	RTC auto wakeup timer clock is RTC clock divided by 16
WAKEUP_RTCK_DIV 8	RTC auto wakeup timer clock is RTC clock divided by 8
WAKEUP_RTCK_DIV 4	RTC auto wakeup timer clock is RTC clock divided by 4
WAKEUP_RTCK_DIV 2	RTC auto wakeup timer clock is RTC clock divided by 2
WAKEUP_CKSPRE	RTC auto wakeup timer clock is ckspre
WAKEUP_CKSPRE_2 EXP16	RTC auto wakeup timer clock is ckspre and wakeup timer add 2exp16
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* RTC auto wakeup timer clock is ckspre */
```

```
ErrStatus error_status = rtc_wakeup_clock_set(WAKEUP_RTCK_DIV8);
```

## rtc\_wakeup\_timer\_set

The description of rtc\_wakeup\_timer\_set is shown as below:

**Table 3-1020. Function rtc\_wakeup\_timer\_set**

<b>Function name</b>	rtc_wakeup_timer_set
<b>Function prototype</b>	ErrStatus rtc_wakeup_timer_set(uint16_t wakeup_timer);
<b>Function descriptions</b>	set wakeup timer value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wakeup_timer</b>	wakeup timer value

<i>uint16_t</i>	0x0000-0xffff
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* set wakeup timer value */
```

```
ErrStatus error_status = rtc_wakeup_timer_set(0XFFEE);
```

### rtc\_wakeup\_timer\_get

The description of rtc\_wakeup\_timer\_set is shown as below:

**Table 3-1021. Function rtc\_wakeup\_timer\_get**

<b>Function name</b>	rtc_wakeup_timer_get
<b>Function prototype</b>	uint16_t rtc_wakeup_timer_get(void);
<b>Function descriptions</b>	set wakeup timer value
<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>	0-0xFFFF

Example:

```
/* get wakeup timer value */
```

```
uint32_t wakeup_time = rtc_wakeup_timer_get();
```

### rtc\_smooth\_calibration\_config

The description of rtc\_smooth\_calibration\_config is shown as below:

**Table 3-1022. Function rtc\_smooth\_calibration\_config**

<b>Function name</b>	rtc_smooth_calibration_config
<b>Function prototype</b>	ErrStatus rtc_smooth_calibration_config(uint32_t window, uint32_t plus, uint32_t minus);
<b>Function descriptions</b>	configure RTC smooth calibration
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>window</b>	select calibration window

<i>RTC_CALIBRATION_WINDOW_32S</i>	2exp20 RTCCLK cycles, 32s if RTCCLK = 32768 Hz
<i>RTC_CALIBRATION_WINDOW_16S</i>	2exp19 RTCCLK cycles, 16s if RTCCLK = 32768 Hz
<i>RTC_CALIBRATION_WINDOW_8S</i>	2exp18 RTCCLK cycles, 8s if RTCCLK = 32768 Hz
<b>Input parameter{in}</b>	
<b>plus</b>	add RTC clock or not
<i>RTC_CALIBRATION_PLUS_SET</i>	add one RTC clock every 2048 rtc clock
<i>RTC_CALIBRATION_PLUS_RESET</i>	no effect
<b>Input parameter{in}</b>	
<b>minus</b>	the RTC clock to minus during the calibration window(0x0 - 0x1FF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>ErrStatus</b>	ERROR or SUCCESS

Example:

```
/* configure RTC smooth calibration */
```

```
ErrStatus error_status;
```

```
error_status = rtc_smooth_calibration_config(RTC_CALIBRATION_WINDOW_32S,
RTC_CALIBRATION_PLUS_SET, 0x10);
```

### rtc\_interrupt\_enable

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

**Table 3-1023. Function rtc\_interrupt\_enable**

<b>Function name</b>	rtc_interrupt_enable
<b>Function prototype</b>	void rtc_interrupt_enable(uint32_t interrupt);
<b>Function descriptions</b>	enable specified RTC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	specify which interrupt source to be enabled
<i>RTC_INT_TIMESTAMP</i>	timestamp interrupt
<i>RTC_INT_ALARM0</i>	Alarm0 interrupt
<i>RTC_INT_ALARM1</i>	Alarm1 interrupt
<i>RTC_INT_TAMP_ALL</i>	tamp interrupt
<i>RTC_INT_TAMP0</i>	Tamper0 detection interrupt
<i>RTC_INT_TAMP1</i>	Tamper1 detection interrupt

<i>RTC_INT_TAMP2</i>	Tamper2 detection interrupt
<i>RTC_INT_WAKEUP</i>	wakeup timer interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable specified RTC interrupt*/
rtc_interrupt_enable(RTC_INT_TAMP0);
```

### rtc\_interrupt\_disable

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

**Table 3-1024. Function rtc\_interrupt\_disable**

<b>Function name</b>	rtc_interrupt_disable
<b>Function prototype</b>	void rtc_interrupt_disable(uint32_t interrupt);
<b>Function descriptions</b>	disble specified RTC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>interrupt</b>	specify which RTC interrupt to disable
<i>RTC_INT_TIMESTAMP</i>	timestamp interrupt
<i>RTC_INT_ALARM0</i>	Alarm0 interrupt
<i>RTC_INT_ALARM1</i>	Alarm1 interrupt
<i>RTC_INT_TAMP_ALL</i>	All tamp interrupt
<i>RTC_INT_TAMP0</i>	Tamper0 detection interrupt
<i>RTC_INT_TAMP1</i>	Tamper1 detection interrupt
<i>RTC_INT_TAMP2</i>	Tamper2 detection interrupt
<i>RTC_INT_WAKEUP</i>	wakeup timer interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disble specified RTC interrupt */
rtc_interrupt_disable(RTC_INT_TAMP0);
```

### rtc\_flag\_get

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

Table 3-1025. Function rtc\_flag\_get

Function name	rtc_flag_get
Function prototype	FlagStatus rtc_flag_get(uint32_t flag);
Function descriptions	check specified flag
Precondition	-
The called functions	-
Input parameter{in}	
flag	specify which flag to check
RTC_FLAG_SCP	smooth calibration pending flag
RTC_FLAG_TP2	RTC tamper 2 detected flag
RTC_FLAG_TP1	RTC tamper 1 detected flag
RTC_FLAG_TP0	RTC tamper 0 detected flag
RTC_FLAG_TSOVR	time-stamp overflow event flag
RTC_FLAG_TS	time-stamp event flag
RTC_FLAG_ALARM0	Alarm0 event flag
RTC_FLAG_ALARM1	Alarm1 event flag
RTC_FLAG_WT	wakeup timer occurs flag
RTC_FLAG_INIT	init mode event flag
RTC_FLAG_RSYN	time and date registers synchronized event flag
RTC_FLAG_YCM	year parameter configured event flag
RTC_FLAG_SOP	shift function operation pending flag
RTC_FLAG_ALARM0 W	Alarm0 written available flag
RTC_FLAG_ALARM1 W	Alarm1 written available flag
RTC_FLAG_WTW	wakeup timer can be written flag
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* check time-stamp event flag */
```

```
FlagStatus = rtc_flag_get(RTC_FLAG_TS);
```

### rtc\_flag\_clear

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

Table 3-1026. Function rtc\_flag\_clear

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

The called functions	-
Input parameter{in}	
flag	specify which flag to clear
RTC_FLAG_TP1	RTC tamper 1 detected flag
RTC_FLAG_TP0	RTC tamper 0 detected flag
RTC_FLAG_TSOVR	time-stamp overflow event flag
RTC_FLAG_TS	time-stamp event flag
RTC_FLAG_WT	wakeup timer occurs flag
RTC_FLAG_ALARM0	Alarm0 event flag
RTC_FLAG_ALARM1	Alarm1 event flag
RTC_FLAG_RSYN	time and date registers synchronized event flag
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear time-stamp event flag */
rtc_flag_clear(RTC_FLAG_TS);
```

## 3.30. SPI

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S audio protocol. The SPI/I2S registers are listed in chapter [3.30.1](#), the SPI/I2S firmware functions are introduced in chapter [3.30.2](#).

### 3.30.1. Descriptions of Peripheral registers

SPI/I2S registers are listed in the table shown as below:

**Table 3-1027. SPI/I2S Registers**

Registers	Descriptions
SPI_CTL0	Control register 0
SPI_CTL1	Control register 1
SPI_CFG0	Configuration register 0
SPI_CFG1	Configuration register 1
SPI_INT	Interrupt register
SPI_STAT	Status register
SPI_STATC	Interrupt/Status flags clear register
SPI_TDATA	Data Transfer register
SPI_RDARA	Data Receive register
SPI_CRCPOLY	CRC polynomial register
SPI_TCRC	TX CRC register

Registers	Descriptions
SPI_RCRC	RX CRC register
SPI_URDATA	Underrun Data register
SPI_I2SCTL	I2S control register
SPI_QCTL	Quad_SPI mode control register
SPI_RXDLYCK	RX clock delay register

### 3.30.2. Descriptions of Peripheral functions

SPI/I2S firmware functions are listed in the table shown as below:

**Table 3-1028. SPI/I2S firmware function**

Function name	Function description
spi_i2s_deinit	reset SPI and I2S peripheral
spi_struct_para_init	initialize the parameters of SPI struct with the default values
spi_init	initialize SPI peripheral parameter
spi_enable	enable SPI
spi_disable	disable SPI
i2s_init	initialize I2S peripheral parameter
i2s_psc_config	configure I2S peripheral prescaler
i2s_enable	enable I2S
i2s_disable	disable I2S
spi_io_config	SPI MOSI and MISO pin swap
spi_nss_idleness_delay_set	set delay between active edge of NSS and start transfer or receive data in SPI master mode
spi_data_frame_delay_set	set SPI master data frame delay
spi_master_receive_clock_delay_set	set SPI master mode rx clock delay
spi_slave_receive_clock_delay_set	set SPI slave mode rx clock delay
spi_master_receive_clock_delay_clear	clear SPI master mode rx clock delay
spi_slave_receive_clock_delay_clear	clear SPI slave mode rx clock delay
spi_nss_output_control	SPI NSS output control
spi_nss_polarity_set	set SPI NSS active polarity
spi_nss_output_enable	enable SPI NSS output
spi_nss_output_disable	disable SPI NSS output
spi_nss_internal_high	SPI NSS pin high level in software mode
spi_nss_internal_low	SPI NSS pin low level in software mode
spi_dma_enable	enable SPI DMA send or receive
spi_dma_disable	disable SPI DMA send or receive
spi_i2s_data_frame_size_config	configure SPI/I2S data frame size
spi_i2s_data_transmit	SPI/I2S transmit data
spi_i2s_data_receive	SPI/I2S receive data
spi_bidirectional_transfer_config	configure SPI bidirectional transfer direction
spi_master_transfer_start	SPI/I2S master start transfer

Function name	Function description
spi_current_data_num_config	configure SPI current data number
spi_reload_data_num_config	configure SPI reload data number
spi_crc_polynomial_set	set SPI CRC polynomial
spi_crc_polynomial_get	get SPI CRC polynomial
spi_crc_length_config	configure SPI CRC length
spi_crc_on	turn on CRC function
spi_crc_off	turn off CRC function
spi_crc_get	get SPI CRC send value or receive value
spi_crc_full_size_enable	enable SPI CRC full size(33 bit or 17 bit) polynomial
spi_crc_full_size_disable	disable SPI CRC full size(33 bit or 17 bit) polynomial
spi_tcr_init_pattern	configure SPI TCRC init pattern
spi_rcrc_init_pattern	configure SPI RCRC init pattern
spi_ti_mode_enable	enable SPI TI mode
spi_ti_mode_disable	disable SPI TI mode
spi_quad_enable	enable quad wire SPI
spi_quad_disable	disable quad wire SPI
spi_quad_write_enable	enable quad wire SPI write
spi_quad_read_enable	enable quad wire SPI read
spi_quad_io23_output_enable	enable quad wire SPI_IO2 and SPI_IO3 pin output
spi_quad_io23_output_disable	disable quad wire SPI_IO2 and SPI_IO3 pin output
spi_underrun_operation	slave transmitter underrun detected operation
spi_underrun_config	configure slave transmitter underrun detected
spi_underrun_data_config	configure underrun data at slave mode
spi_suspend_mode_config	configure SPI suspend in receive mode
spi_suspend_request	SPI master mode suspend request
spi_related_ios_af_enable	enable SPI related IOs AF
spi_related_ios_af_disable	disable SPI related IOs AF
spi_af_gpio_control	SPI af gpio control
spi_i2s_interrupt_enable	enable SPI and I2S interrupt
spi_i2s_interrupt_disable	disable SPI and I2S interrupt
spi_i2s_interrupt_flag_get	get SPI and I2S interrupt flag status
spi_i2s_flag_get	get SPI and I2S flag status
spi_i2s_flag_clear	clear SPI and I2S flag status
spi_i2s_rxfifo_plevel_get	get SPI and I2S RXFIFO packing level
spi_i2s_remain_data_num_get	get SPI and I2S remaining data frames number in the TXSIZE session
spi_fifo_threshold_level_set	set SPI FIFO threshold level
spi_word_access_enable	enable SPI word access
spi_word_access_disable	disable SPI word access
spi_byte_access_enable	enable SPI byte access
spi_byte_access_disable	disable SPI byte access



## Structure spi\_parameter\_struct

**Table 3-1029. 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_DATASIZE_xBIT, x=4,5,...32)
nss	SPI NSS control by hardware or software (SPI_NSS_SOFT, SPI_NSS_HARD)
endian	SPI big endian or little endian (SPI_ENDIAN_MSB, SPI_ENDIAN_LSB)
clock_polarity_phase	SPI clock phase and polarity (SPI_CK_PL_LOW_PH_1EDGE, SPI_CK_PL_HIGH_PH_1EDGE, SPI_CK_PL_LOW_PH_2EDGE, SPI_CK_PL_HIGH_PH_2EDGE)
prescale	SPI prescale factor (SPI_PSC_n (n=2,4,8,16,32,64,128,256))

## spi\_i2s\_deinit

The description of spi\_i2s\_deinit is shown as below:

**Table 3-1030. Function spi\_i2s\_deinit**

<b>Function name</b>	spi_i2s_deinit
<b>Function prototype</b>	void spi_i2s_deinit(uint32_t spi_periph);
<b>Function descriptions</b>	reset SPI and I2S peripheral
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI/I2S peripheral
<b>SPIx</b>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset SPI0 */
spi_i2s_deinit(SPI0);
```

## spi\_struct\_para\_init

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

**Table 3-1031. Function spi\_struct\_para\_init**

<b>Function name</b>	spi_struct_para_init
<b>Function prototype</b>	void spi_struct_para_init(spi_parameter_struct* spi_struct);
<b>Function descriptions</b>	initialize the parameters of SPI struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
<b>spi_struct</b>	SPI parameter struct, the structure members can refer to members of the structure <a href="#">Table 3-1029. spi_parameter_struct</a>
<b>Return value</b>	
-	-

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 3-1032. Function spi\_init**

<b>Function name</b>	spi_init
<b>Function prototype</b>	void spi_init(uint32_t spi_periph, spi_parameter_struct* spi_struct);
<b>Function descriptions</b>	initialize SPI peripheral parameter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>spi_struct</b>	SPI parameter initialization struct, the structure members can refer to members of the structure <a href="#">Table 3-1029. spi_parameter_struct</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* initialize SPI0 */

spi_parameter_struct spi_init_struct;

spi_init_struct.trans_mode = SPI_TRANSMODE_BDTRANSMIT;

spi_init_struct.device_mode = SPI_MASTER;

spi_init_struct.frame_size = SPI_FRAME_SIZE_8BIT;

spi_init_struct.clock_polarity_phase = SPI_CK_PL_HIGH_PH_2EDGE;

spi_init_struct.nss = SPI_NSS_SOFT;

spi_init_struct.prescale = SPI_PSC_8;

spi_init_struct.endian = SPI_ENDIAN_MSB;

spi_init(SPI0, &spi_init_struct);

```

### spi\_enable

The description of spi\_enable is shown as below:

**Table 3-1033. Function spi\_enable**

<b>Function name</b>	spi_enable
<b>Function prototype</b>	void spi_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* enable SPI0 */

spi_enable(SPI0);

```

### spi\_disable

The description of spi\_disable is shown as below:

**Table 3-1034. Function spi\_disable**

<b>Function name</b>	spi_disable
<b>Function prototype</b>	void spi_disable(uint32_t spi_periph);

<b>Function descriptions</b>	disable SPIx
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 */
spi_disable(SPI0);
```

## i2s\_init

The description of i2s\_init is shown as below:

**Table 3-1035. Function i2s\_init**

<b>Function name</b>	i2s_init
<b>Function prototype</b>	void i2s_init(uint32_t spi_periph, uint32_t i2s_mode, uint32_t i2s_standard, uint32_t i2s_ckpl);
<b>Function descriptions</b>	initialize I2S peripheral parameter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	I2S peripheral
<i>SPIx</i>	x=0,1,2,5
<b>Input parameter{in}</b>	
<b>i2s_mode</b>	I2S operation mode
<i>I2S_MODE_SLAVETX</i>	I2S slave transmit mode
<i>I2S_MODE_SLAVRX</i>	I2S slave receive mode
<i>I2S_MODE_MASTERTX</i>	I2S master transmit mode
<i>I2S_MODE_MASTERR</i>	I2S master receive mode
<b>Input parameter{in}</b>	
<b>i2s_standard</b>	I2S standard
<i>I2S_STD_PHILLIPS</i>	I2S phillips standard
<i>I2S_STD_MSB</i>	I2S MSB standard
<i>I2S_STD_LSB</i>	I2S LSB standard
<i>I2S_STD_PCMSHORT</i>	I2S PCM short standard
<i>I2S_STD_PCMLONG</i>	I2S PCM long standard

Input parameter{in}	
<b>i2s_ckpl</b>	I2S idle state clock polarity
<i>I2S_CKPL_LOW</i>	I2S clock polarity low level
<i>I2S_CKPL_HIGH</i>	I2S clock polarity high level
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize I2S0 */
```

```
i2s_init(SPI0, I2S_MODE_MASTERTX, I2S_STD_PHILLIPS, I2S_CKPL_LOW);
```

### i2s\_psc\_config

The description of i2s\_psc\_config is shown as below:

**Table 3-1036. Function i2s\_psc\_config**

<b>Function name</b>	i2s_psc_config
<b>Function prototype</b>	void i2s_psc_config(uint32_t spi_periph, uint32_t i2s_audiosample, uint32_t i2s_frameformat, uint32_t i2s_mckout);
<b>Function descriptions</b>	configure I2S prescaler
<b>Precondition</b>	-
<b>The called functions</b>	rcu_clock_freq_get
Input parameter{in}	
<b>spi_periph</b>	I2S peripheral
<i>SPIx</i>	x=0,1,2,5
Input parameter{in}	
<b>i2s_audiosample</b>	I2S audio sample rate
<i>I2S_AUDIOSAMPLE_8K</i>	audio sample rate is 8KHz
<i>I2S_AUDIOSAMPLE_11K</i>	audio sample rate is 11KHz
<i>I2S_AUDIOSAMPLE_16K</i>	audio sample rate is 16KHz
<i>I2S_AUDIOSAMPLE_22K</i>	audio sample rate is 22KHz
<i>I2S_AUDIOSAMPLE_32K</i>	audio sample rate is 32KHz
<i>I2S_AUDIOSAMPLE_44K</i>	audio sample rate is 44KHz
<i>I2S_AUDIOSAMPLE_48K</i>	audio sample rate is 48KHz
<i>I2S_AUDIOSAMPLE_96K</i>	audio sample rate is 96KHz

6K	
I2S_AUDIOSAMPLE_1 92K	audio sample rate is 192KHz
<b>Input parameter{in}</b>	
<b>i2s_frameformat</b>	I2S data length and channel length
I2S_FRAMEFORMAT_ DT16B_CH16B	I2S data length is 16 bit and channel length is 16 bit
I2S_FRAMEFORMAT_ DT16B_CH32B	I2S data length is 16 bit and channel length is 32 bit
I2S_FRAMEFORMAT_ DT24B_CH32B	I2S data length is 24 bit and channel length is 32 bit
I2S_FRAMEFORMAT_ DT32B_CH32B	I2S data length is 32 bit and channel length is 32 bit
<b>Input parameter{in}</b>	
<b>i2s_mckout</b>	I2S master clock output
I2S_MCKOUT_ENABL E	I2S master clock output enable
I2S_MCKOUT_DISABL E	I2S master clock output disable
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure I2S0 prescaler */
```

```
i2s_psc_config(SPI0, I2S_AUDIOSAMPLE_44K, I2S_FRAMEFORMAT_DT16B_CH16B,  
I2S_MCKOUT_DISABLE);
```

### i2s\_enable

The description of i2s\_enable is shown as below:

**Table 3-1037. Function i2s\_enable**

<b>Function name</b>	i2s_enable
<b>Function prototype</b>	void i2s_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable I2S
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	I2S peripheral
SPIx	x=0,1,2,5
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* enable I2S0*/
i2s_enable(SPI0);
```

### i2s\_disable

The description of i2s\_disable is shown as below:

**Table 3-1038. Function i2s\_disable**

Function name	i2s_disable
Function prototype	void i2s_disable(uint32_t spi_periph);
Function descriptions	disable I2S
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	I2S peripheral
SPIx	x=0,1,2,5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2S0*/
i2s_disable(SPI0);
```

### spi\_io\_config

The description of spi\_io\_config is shown as below:

**Table 3-1039. Function spi\_io\_config**

Function name	spi_io_config
Function prototype	void spi_io_config(uint32_t spi_periph, uint32_t io_cfg);
Function descriptions	SPI MOSI and MISO pin swap
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Input parameter{in}	
io_cfg	SPI IO swap config

<i>SPI_IO_SWAP</i>	SPI MOSI and MISO swap
<i>SPI_IO_NORMAL</i>	SPI MOSI and MISO no swap
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI0 MOSI and MISO pin swap */
spi_io_config(SPI0, SPI_IO_SWAP);
```

### spi\_nss\_idleness\_delay\_set

The description of spi\_nss\_idleness\_delay\_set is shown as below:

**Table 3-1040. Function spi\_nss\_idleness\_delay\_set**

<b>Function name</b>	spi_nss_idleness_delay_set
<b>Function prototype</b>	void spi_nss_idleness_delay_set(uint32_t spi_periph, uint32_t delay_cycle);
<b>Function descriptions</b>	set delay between active edge of NSS and start transfer or receive data in SPI master mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>delay_cycle</b>	delay cycle
<i>SPI_NSS_IDLENESS_xCYCLE</i>	x clock cycle delay between active edge of NSS and transmission (x = 00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set SPI0 1 cycle nss idleness delay */
spi_nss_idleness_delay_set(SPI0, SPI_NSS_IDLENESS_01CYCLE);
```

### spi\_data\_frame\_delay\_set

The description of spi\_data\_frame\_delay\_set is shown as below:

**Table 3-1041. Function spi\_data\_frame\_delay\_set**

<b>Function name</b>	spi_data_frame_delay_set
----------------------	--------------------------



<b>Function prototype</b>	void spi_data_frame_delay_set(uint32_t spi_periph, uint32_t delay_cycle);
<b>Function descriptions</b>	set SPI master data frame delay
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>delay_cycle</b>	delay cycle
<i>SPI_DATA_IDLENESS_xCYCLE</i>	x clock cycle delay between data frames in SPI master mode (x = 00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set SPI0 1 cycle data frame delay */
```

```
spi_data_frame_delay_set(SPI0, SPI_DATA_IDLENESS_01CYCLE);
```

### spi\_master\_receive\_clock\_delay\_set

The description of spi\_master\_receive\_clock\_delay\_set is shown as below:

**Table 3-1042. Function spi\_master\_receive\_clock\_delay\_set**

<b>Function name</b>	spi_master_receive_clock_delay_set
<b>Function prototype</b>	void spi_master_receive_clock_delay_set(uint32_t spi_periph, uint32_t delay_unit);
<b>Function descriptions</b>	set SPI master mode rx clock delay
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>delay_unit</b>	clock delay unit(0-0x1F)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set SPI0 master mode rx clock delay 16 units */
```

```
spi_master_receive_clock_delay_set(SPI0, 0x0F);
```

### spi\_slave\_receive\_clock\_delay\_set

The description of spi\_slave\_receive\_clock\_delay\_set is shown as below:

**Table 3-1043. Function spi\_slave\_receive\_clock\_delay\_set**

<b>Function name</b>	spi_slave_receive_clock_delay_set
<b>Function prototype</b>	void spi_slave_receive_clock_delay_set(uint32_t spi_periph, uint32_t delay_unit);
<b>Function descriptions</b>	set SPI slave mode rx clock delay
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>delay_unit</b>	clock delay unit(0-0x1F)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set SPI0 slave mode rx clock delay 16 units */
```

```
spi_slave_receive_clock_delay_set(SPI0, 0x0F);
```

### spi\_master\_receive\_clock\_delay\_clear

The description of spi\_master\_receive\_clock\_delay\_clear is shown as below:

**Table 3-1044. Function spi\_master\_receive\_clock\_delay\_clear**

<b>Function name</b>	spi_master_receive_clock_delay_clear
<b>Function prototype</b>	void spi_master_receive_clock_delay_clear(uint32_t spi_periph);
<b>Function descriptions</b>	clear SPI master mode rx clock delay
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear SPI0 master mode rx clock delay */

spi_master_receive_clock_delay_clear(SPI0);
```

### spi\_slave\_receive\_clock\_delay\_clear

The description of spi\_slave\_receive\_clock\_delay\_clear is shown as below:

**Table 3-1045. Function spi\_slave\_receive\_clock\_delay\_clear**

<b>Function name</b>	spi_slave_receive_clock_delay_clear
<b>Function prototype</b>	void spi_slave_receive_clock_delay_clear(uint32_t spi_periph);
<b>Function descriptions</b>	clear SPI slave mode rx clock delay
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear SPI0 slave mode rx clock delay */

spi_slave_receive_clock_delay_clear(SPI0);
```

### spi\_nss\_output\_control

The description of spi\_nss\_output\_control is shown as below:

**Table 3-1046. Function spi\_nss\_output\_control**

<b>Function name</b>	spi_nss_output_control
<b>Function prototype</b>	void spi_nss_output_control(uint32_t spi_periph, uint32_t nss_ctl);
<b>Function descriptions</b>	SPI NSS output control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>nss_ctl</b>	nss control bit
<i>SPI_NSS_HOLD_UNTIL_TRANS_END</i>	SPI NSS remains active level until data transfer complete
<i>SPI_NSS_INVALID_PULSES</i>	SPI data frames are interleaved with NSS invalid pulses

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

Example:

```
/* set SPI0 NSS hold until trans end */
```

```
spi_nss_output_control(SPI0, SPI_NSS_HOLD_UNTIL_TRANS_END);
```

### spi\_nss\_polarity\_set

The description of spi\_nss\_polarity\_set is shown as below:

**Table 3-1047. Function spi\_nss\_polarity\_set**

<b>Function name</b>	spi_nss_polarity_set
<b>Function prototype</b>	void spi_nss_polarity_set(uint32_t spi_periph, uint32_t polarity);
<b>Function descriptions</b>	set SPI NSS active polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>polarity</b>	SPI NSS active level
<i>SPI_NSS_POLARITY_HIGH</i>	SPI NSS high level is active
<i>SPI_NSS_POLARITY_LOW</i>	SPI NSS low level is active
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set SPI0 NSS high level is active */
```

```
spi_nss_polarity_set(SPI0, SPI_NSS_POLARITY_HIGH);
```

### spi\_nss\_output\_enable

The description of spi\_nss\_output\_enable is shown as below:

**Table 3-1048. Function spi\_nss\_output\_enable**

<b>Function name</b>	spi_nss_output_enable
----------------------	-----------------------

<b>Function prototype</b>	void spi_nss_output_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI NSS output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI0 NSS output */
spi_nss_output_enable(SPI0);
```

### spi\_nss\_output\_disable

The description of spi\_nss\_output\_disable is shown as below:

**Table 3-1049. Function spi\_nss\_output\_disable**

<b>Function name</b>	spi_nss_output_disable
<b>Function prototype</b>	void spi_nss_output_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI NSS output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 NSS output */
spi_nss_output_disable(SPI0);
```

### spi\_nss\_internal\_high

The description of spi\_nss\_internal\_high is shown as below:

**Table 3-1050. Function spi\_nss\_internal\_high**

<b>Function name</b>	spi_nss_internal_high
----------------------	-----------------------

<b>Function prototype</b>	void spi_nss_internal_high(uint32_t spi_periph);
<b>Function descriptions</b>	SPI NSS pin high level in software mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI0 NSS pin is pulled high level in software mode */
```

```
spi_nss_internal_high(SPI0);
```

### spi\_nss\_internal\_low

The description of spi\_nss\_internal\_low is shown as below:

**Table 3-1051. Function spi\_nss\_internal\_low**

<b>Function name</b>	spi_nss_internal_low
<b>Function prototype</b>	void spi_nss_internal_low(uint32_t spi_periph);
<b>Function descriptions</b>	SPI NSS pin low level in software mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI0 NSS pin is pulled low level in software mode */
```

```
spi_nss_internal_low(SPI0);
```

### spi\_dma\_enable

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

**Table 3-1052. Function spi\_dma\_enable**

<b>Function name</b>	spi_dma_enable
----------------------	----------------

<b>Function prototype</b>	void spi_dma_enable(uint32_t spi_periph, uint8_t spi_dma);
<b>Function descriptions</b>	enable SPI DMA send or receive
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>spi_dma</b>	SPI DMA mode
<i>SPI_DMA_TRANSMIT</i>	SPI transmit data use DMA
<i>SPI_DMA_RECEIVE</i>	SPI receive data use DMA
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI0 transmit data DMA function */
spi_dma_enable(SPI0, SPI_DMA_TRANSMIT);
```

### spi\_dma\_disable

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

**Table 3-1053. Function spi\_dma\_disable**

<b>Function name</b>	spi_dma_disable
<b>Function prototype</b>	void spi_dma_disable(uint32_t spi_periph, uint8_t spi_dma);
<b>Function descriptions</b>	disable SPI DMA function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>spi_dma</b>	SPI DMA mode
<i>SPI_DMA_TRANSMIT</i>	SPI transmit data use DMA
<i>SPI_DMA_RECEIVE</i>	SPI receive data use DMA
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 transmit data DMA function */
```

```
spi_dma_disable(SPI0, SPI_DMA_TRANSMIT);
```

### spi\_i2s\_data\_frame\_size\_config

The description of spi\_i2s\_data\_frame\_size\_config is shown as below:

**Table 3-1054. Function spi\_i2s\_data\_frame\_size\_config**

<b>Function name</b>	spi_i2s_data_frame_size_config
<b>Function prototype</b>	void spi_i2s_data_frame_size_config(uint32_t spi_periph, uint32_t frame_size);
<b>Function descriptions</b>	configure SPI/I2S data frame size
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>frame_size</b>	SPI frame size
<i>SPI_DATASIZE_xBIT</i>	SPI x-bit data frame size (x=4,5,..32)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure SPI0/I2S0 data frame size is 16 bits */
```

```
spi_i2s_data_frame_size_config(SPI0, SPI_DATASIZE_16BIT);
```

### spi\_i2s\_data\_transmit

The description of spi\_i2s\_data\_transmit is shown as below:

**Table 3-1055. Function spi\_i2s\_data\_transmit**

<b>Function name</b>	spi_i2s_data_transmit
<b>Function prototype</b>	void spi_i2s_data_transmit(uint32_t spi_periph, uint32_t data);
<b>Function descriptions</b>	SPI transmit data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>data</b>	32-bit data
<b>Output parameter{out}</b>	
-	-



Return value	
-	-

Example:

```
/* SPI0 transmit data */
```

```
spl_i2s_data_transmit(SPI0, spl0_send_array[send_n]);
```

### spl\_i2s\_data\_receive

The description of spl\_i2s\_data\_receive is shown as below:

**Table 3-1056. Function spl\_i2s\_data\_receive**

Function name	spl_i2s_data_receive
Function prototype	Uln32_t spl_i2s_data_receive(uIn32_t spl_periph);
Function descriptions	SPI receive data
Precondition	-
The called functions	-
Input parameter{in}	
spl_periph	SPI peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
uIn32_t	32-bit data

Example:

```
/* SPI0 receive data */
```

```
spl0_receive_array[receive_n] = spl_i2s_data_receive(SPI0);
```

### spl\_bidirectional\_transfer\_config

The description of spl\_bidirectional\_transfer\_config is shown as below:

**Table 3-1057. Function spl\_bidirectional\_transfer\_config**

Function name	spl_bidirectional_transfer_config
Function prototype	void spl_bidirectional_transfer_config(uIn32_t spl_periph, uIn32_t transfer_direction);
Function descriptions	configure SPI bidirectional transfer direction
Precondition	-
The called functions	-
Input parameter{in}	
spl_periph	SPI peripheral
SPIx	x=0,1...5
Input parameter{in}	

<b>transfer_direction</b>	SPI transfer direction
<i>SPI_BIDIRECTIONAL_TRANSMIT</i>	SPI work in transmit-only mode
<i>SPI_BIDIRECTIONAL_RECEIVE</i>	SPI work in receive-only mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI0 works in transmit-only mode */
```

```
spi_bidirectional_transfer_config(SPI0, SPI_BIDIRECTIONAL_TRANSMIT);
```

### spi\_master\_transfer\_start

The description of spi\_master\_transfer\_start is shown as below:

**Table 3-1058. Function spi\_master\_transfer\_start**

<b>Function name</b>	spi_master_transfer_start
<b>Function prototype</b>	void spi_master_transfer_start(uint32_t spi_periph, uint32_t transfer_start);
<b>Function descriptions</b>	SPI/I2S master start transfer
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>transfer_start</b>	transfer start bit
<i>SPI_TRANS_START</i>	the master transmission is occurring, or has been temporarily suspended by automatic suspend
<i>SPI_TRANS_IDLE</i>	the master transfer is idle status
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI0 master transfer start */
```

```
spi_master_transfer_start(SPI0, SPI_TRANS_START);
```

### spi\_current\_data\_num\_config

The description of spi\_current\_data\_num\_config is shown as below:

Table 3-1059. Function spi\_current\_data\_num\_config

Function name	spi_current_data_num_config
Function prototype	void spi_current_data_num_config(uint32_t spi_periph, uint32_t current_num);
Function descriptions	configure SPI current data number
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Input parameter{in}	
current_num	SPI transfer current data number (0-0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 transfer current data number */
spi_current_data_num_config(SPI0, spi0_current_array[current_n]);
```

### spi\_reload\_data\_num\_config

The description of spi\_reload\_data\_num\_config is shown as below:

Table 3-1060. Function spi\_reload\_data\_num\_config

Function name	spi_reload_data_num_config
Function prototype	void spi_reload_data_num_config(uint32_t spi_periph, uint32_t reload_num)
Function descriptions	configure SPI reload data number
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Input parameter{in}	
reload_num	SPI transfer reload data number (0-0xFFFF)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* SPI0 transfer reload data number */
```

```
spi_reload_data_num_config(SPI0, spi0_reload_array[reload_n]);
```

### spi\_crc\_polynomial\_set

The description of spi\_crc\_polynomial\_set is shown as below:

**Table 3-1061. Function spi\_crc\_polynomial\_set**

<b>Function name</b>	spi_crc_polynomial_set
<b>Function prototype</b>	void spi_crc_polynomial_set(uint32_t spi_periph, uint32_t crc_poly);
<b>Function descriptions</b>	set SPI CRC polynomial
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>crc_poly</b>	CRC polynomial value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set SPI0 CRC polynomial */
spi_crc_polynomial_set(SPI0,CRC_VALUE);
```

### spi\_crc\_polynomial\_get

The description of spi\_crc\_polynomial\_get is shown as below:

**Table 3-1062. Function spi\_crc\_polynomial\_get**

<b>Function name</b>	spi_crc_polynomial_get
<b>Function prototype</b>	uint32_t spi_crc_polynomial_get(uint32_t spi_periph);
<b>Function descriptions</b>	get SPI CRC polynomial
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	32-bit CRC polynomial (0-0xFFFFFFFF)

Example:

```

/* get SPI0 CRC polynomial */

uint32_t crc_val;

crc_val = spi_crc_polynomial_get(SPI0);

```

### spi\_crc\_length\_config

The description of spi\_crc\_length\_config is shown as below:

**Table 3-1063. Function spi\_crc\_length\_config**

<b>Function name</b>	spi_crc_length_config
<b>Function prototype</b>	void spi_crc_length_config(uint32_t spi_periph, uint32_t crc_size);
<b>Function descriptions</b>	configure SPI CRC length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>crc_size</b>	crc size
<i>SPI_CRCSIZE_xBIT</i>	SPI x-bit crc size (x = 4,5,6...32)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* config SPI0 CRC 16-bit length */

spi_crc_length_config(SPI0, SPI_CRCSIZE_16BIT);

```

### spi\_crc\_on

The description of spi\_crc\_on is shown as below:

**Table 3-1064. Function spi\_crc\_on**

<b>Function name</b>	spi_crc_on
<b>Function prototype</b>	void spi_crc_on(uint32_t spi_periph);
<b>Function descriptions</b>	turn on CRC function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* turn on SPI0 CRC function */
```

```
spi_crc_on(SPI0);
```

### spi\_crc\_off

The description of spi\_crc\_off is shown as below:

**Table 3-1065. Function spi\_crc\_off**

Function name	spi_crc_off
Function prototype	void spi_crc_off(uint32_t spi_periph);
Function descriptions	turn off CRC function
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* turn off SPI0 CRC function */
```

```
spi_crc_off(SPI0);
```

### spi\_crc\_get

The description of spi\_crc\_get is shown as below:

**Table 3-1066. Function spi\_crc\_get**

Function name	spi_crc_get
Function prototype	uint32_t spi_crc_get(uint32_t spi_periph, uint8_t crc);
Function descriptions	get SPI CRC send value or receive value
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Input parameter{in}	
crc	SPI crc value

<i>SPI_CRC_TX</i>	get transmit CRC value
<i>SPI_CRC_RX</i>	get receive CRC value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	32-bit CRC value (0-0xFFFFFFFF)

Example:

```
/* get SPI0 CRC send value */

uint32_t value;

value = spi_crc_get(SPI0, SPI_CRC_TX);
```

### spi\_crc\_full\_size\_enable

The description of spi\_crc\_full\_size\_enable is shown as below:

**Table 3-1067. Function spi\_crc\_full\_size\_enable**

<b>Function name</b>	spi_crc_full_size_enable
<b>Function prototype</b>	void spi_crc_full_size_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI CRC full size(33 bit or 17 bit) polynomial
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

Example:

```
/* enable SPI0 crc full size */

spi_crc_full_size_enable(SPI0);
```

### spi\_crc\_full\_size\_disable

The description of spi\_crc\_full\_size\_disable is shown as below:

**Table 3-1068. Function spi\_crc\_full\_size\_disable**

<b>Function name</b>	spi_crc_full_size_disable
<b>Function prototype</b>	void spi_crc_full_size_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI CRC full size(33 bit or 17 bit) polynomial
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
Output parameter{out}	
-	-
Return value	

Example:

```
/* disable SPI0 crc full size */
```

```
spi_crc_full_size_disable (SPI0);
```

### spi\_tcrcl\_init\_pattern

The description of spi\_tcrcl\_init\_pattern is shown as below:

**Table 3-1069. Function spi\_tcrcl\_init\_pattern**

<b>Function name</b>	spi_tcrcl_init_pattern
<b>Function prototype</b>	void spi_tcrcl_init_pattern(uint32_t spi_periph, uint32_t init_pattern);
<b>Function descriptions</b>	configure SPI TCRC init pattern
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
Input parameter{in}	
<b>init_pattern</b>	SPI crc value
<i>SPI_TCRC_INIT_1</i>	use all 1 pattern
<i>SPI_TCRC_INIT_0</i>	use all 0 pattern
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* config SPI0 TCRC all 1 initial pattern */
```

```
spi_tcrcl_init_pattern(SPI0, SPI_TCRC_INIT_1);
```

### spi\_rrcl\_init\_pattern

The description of spi\_rrcl\_init\_pattern is shown as below:

**Table 3-1070. Function spi\_rrcl\_init\_pattern**

<b>Function name</b>	spi_rrcl_init_pattern
----------------------	-----------------------



<b>Function prototype</b>	void spi_rcrc_init_pattern(uint32_t spi_periph, uint32_t init_pattern);
<b>Function descriptions</b>	configure SPI RCRC init pattern
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>init_pattern</b>	SPI crc value
<i>SPI_RCRC_INIT_1</i>	use all 1 pattern
<i>SPI_RCRC_INIT_0</i>	use all 0 pattern
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config SPI0 RCRC all 1 initial pattern */
spi_rcrc_init_pattern(SPI0, SPI_RCRC_INIT_1);
```

### spi\_ti\_mode\_enable

The description of spi\_ti\_mode\_enable is shown as below:

**Table 3-1071. Function spi\_ti\_mode\_enable**

<b>Function name</b>	spi_ti_mode_enable
<b>Function prototype</b>	void spi_ti_mode_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI TI mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

Example:

```
/* enable SPI0 TI mode */
spi_ti_mode_enable(SPI0);
```

## spi\_ti\_mode\_disable

The description of spi\_ti\_mode\_disable is shown as below:

**Table 3-1072. Function spi\_ti\_mode\_disable**

<b>Function name</b>	spi_ti_mode_disable
<b>Function prototype</b>	void spi_ti_mode_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI TI mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SP/x</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 TI mode */
spi_ti_mode_disable(SPI0);
```

## spi\_quad\_enable

The description of spi\_quad\_enable is shown as below:

**Table 3-1073. Function spi\_quad\_enable**

<b>Function name</b>	spi_quad_enable
<b>Function prototype</b>	void spi_quad_enable (uint32_t spi_periph);
<b>Function descriptions</b>	enable quad wire SPI
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SP/x</i>	x=3,4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI3 quad wire mode */
spi_quad_enable(SPI3);
```

## spi\_quad\_disable

The description of spi\_quad\_disable is shown as below:

**Table 3-1074. Function spi\_quad\_disable**

<b>Function name</b>	spi_quad_disable
<b>Function prototype</b>	void spi_quad_disable (uint32_t spi_periph);
<b>Function descriptions</b>	disable quad wire SPI
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=3,4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI3 quad wire mode */
spi_quad_disable(SPI3);
```

## spi\_quad\_write\_enable

The description of spi\_quad\_write\_enable is shown as below:

**Table 3-1075. Function spi\_quad\_write\_enable**

<b>Function name</b>	spi_quad_write_enable
<b>Function prototype</b>	void spi_quad_write_enable (uint32_t spi_periph);
<b>Function descriptions</b>	enable quad wire SPI write
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=3,4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI3 quad wire write */
spi_quad_write_enable(SPI3);
```

## spi\_quad\_read\_enable

The description of spi\_quad\_read\_enable is shown as below:

**Table 3-1076. Function spi\_quad\_read\_enable**

<b>Function name</b>	spi_quad_read_enable
<b>Function prototype</b>	void spi_quad_read_enable (uint32_t spi_periph);
<b>Function descriptions</b>	enable quad wire SPI read
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=3,4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI3 quad wire read */
spi_quad_read_enable(SPI3);
```

## spi\_quad\_io23\_output\_enable

The description of spi\_quad\_io23\_output\_enable is shown as below:

**Table 3-1077. Function spi\_quad\_io23\_output\_enable**

<b>Function name</b>	spi_quad_io23_output_enable
<b>Function prototype</b>	void spi_quad_io23_output_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI_IO2 and SPI_IO3 pin output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=3,4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI3 SPI_IO2 and SPI_IO3 pin output */
spi_quad_io23_output_enable(SPI3);
```

## spi\_quad\_io23\_output\_disable

The description of spi\_quad\_io23\_output\_disable is shown as below:

**Table 3-1078. Function spi\_quad\_io23\_output\_disable**

<b>Function name</b>	spi_quad_io23_output_disable
<b>Function prototype</b>	void spi_quad_io23_output_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI_IO2 and SPI_IO3 pin output
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=3,4
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI3 SPI_IO2 and SPI_IO3 pin output */
spi_quad_io23_output_disable(SPI3);
```

## spi\_underrun\_operation

The description of spi\_underrun\_operation is shown as below:

**Table 3-1079. Function spi\_underrun\_operation**

<b>Function name</b>	spi_underrun_operation
<b>Function prototype</b>	void spi_underrun_operation(uint32_t spi_periph, uint32_t ur_ope);
<b>Function descriptions</b>	slave transmitter underrun detected operation
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>ur_ope</b>	underrun operation
<i>SPI_CONFIG_REGISTER_PATTERN</i>	slave send a constant value defined by the SPI_URDATA register
<i>SPI_CONFIG_LAST_RECEIVED</i>	slave send the lastly data frame received from master
<i>SPI_CONFIG_LAST_TRANSMITTED</i>	slave send its lastly transmitted data frame
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* slave underrun detected send a constant value defined by the SPI_URDATA register */
spi_underrun_operation(SPI0, SPI_CONFIG_REGISTER_PATTERN);
```

### spi\_underrun\_config

The description of spi\_underrun\_config is shown as below:

**Table 3-1080. Function spi\_underrun\_config**

<b>Function name</b>	spi_underrun_config
<b>Function prototype</b>	void spi_underrun_config(uint32_t spi_periph, uint32_t ur_cfg);
<b>Function descriptions</b>	configure slave transmitter underrun detected
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>ur_cfg</b>	underrun config
<i>SPI_DETECT_BEGIN_DATA_FRAME</i>	underrun detected at start of data frame (no bit 1 protection)
<i>SPI_DETECT_END_DATA_FRAME</i>	underrun detected at end of last data frame
<i>SPI_DETECT_BEGIN_ACTIVE_NSS</i>	underrun detected at start of NSS signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* slave underrun detected at start of data frame (no bit 1 protection) */
spi_underrun_config(SPI0, SPI_DETECT_BEGIN_DATA_FRAME);
```

### spi\_underrun\_data\_config

The description of spi\_underrun\_data\_config is shown as below:

**Table 3-1081. Function spi\_underrun\_data\_config**

<b>Function name</b>	spi_underrun_data_config
----------------------	--------------------------

<b>Function prototype</b>	void spi_underrun_data_config(uint32_t spi_periph, uint32_t udata);
<b>Function descriptions</b>	configure underrun data at slave mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1...5
<b>Input parameter{in}</b>	
<b>udata</b>	underrun data (0-0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config SPI0 underrun data at slave mode */
spi_underrun_data_config(SPI0, SPI0_URDATA);
```

### spi\_suspend\_mode\_config

The description of spi\_suspend\_mode\_config is shown as below:

**Table 3-1082. Function spi\_suspend\_mode\_config**

<b>Function name</b>	spi_suspend_mode_config
<b>Function prototype</b>	void spi_suspend_mode_config(uint32_t spi_periph, uint32_t sus_mode);
<b>Function descriptions</b>	configure SPI suspend in receive mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1...5
<b>Input parameter{in}</b>	
<b>sus_mode</b>	suspend mode
<b>SPI_AUTO_SUSPEND</b>	until the overrun condition is reached, the SPI stream is suspended in the full RxFIFO state
<b>SPI_CONTINUOUS</b>	SPI stream/clock generation is continuous whether or not an overrun occurs
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config SPI0 auto suspend */
```

```
spi_suspend_mode_config(SPI0, SPI_AUTO_SUSPEND);
```

## spi\_suspend\_request

The description of spi\_suspend\_request is shown as below:

**Table 3-1083. Function spi\_suspend\_request**

<b>Function name</b>	spi_suspend_request
<b>Function prototype</b>	void spi_suspend_request(uint32_t spi_periph);
<b>Function descriptions</b>	SPI master mode suspend request
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI0 master request suspend */
spi_suspend_request(SPI0);
```

## spi\_related\_ios\_af\_enable

The description of spi\_related\_ios\_af\_enable is shown as below:

**Table 3-1084. Function spi\_related\_ios\_af\_enable**

<b>Function name</b>	spi_related_ios_af_enable
<b>Function prototype</b>	void spi_related_ios_af_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI related IOs AF
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

Example:

```
/* enable spi0 related IOs AF */
```



```
spi_related_ios_af_enable(SPI0);
```

## spi\_related\_ios\_af\_disable

The description of spi\_related\_ios\_af\_disable is shown as below:

**Table 3-1085. Function spi\_related\_ios\_af\_disable**

<b>Function name</b>	spi_related_ios_af_disable
<b>Function prototype</b>	void spi_related_ios_af_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI related IOs AF
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable spi0 related IOs AF */
```

```
spi_related_ios_af_disable(SPI0);
```

## spi\_af\_gpio\_control

The description of spi\_af\_gpio\_control is shown as below:

**Table 3-1086. Function spi\_af\_gpio\_control**

<b>Function name</b>	spi_af_gpio_control
<b>Function prototype</b>	void spi_af_gpio_control(uint32_t spi_periph, uint32_t ctl);
<b>Function descriptions</b>	SPI af gpio control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>ctl</b>	gpio control bit
<i>SPI_GPIO_CONTROL</i>	SPI always control all associated GPIO
<i>SPI_GPIO_FREE</i>	SPI do not control GPIO when disabled
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* SPI0 do not control GPIO when disabled */
spi_af_gpio_control(SPI0, SPI_GPIO_FREE);
```

### spi\_i2s\_interrupt\_enable

The description of spi\_i2s\_interrupt\_enable is shown as below:

**Table 3-1087. Function spi\_i2s\_interrupt\_enable**

<b>Function name</b>	spi_i2s_interrupt_enable
<b>Function prototype</b>	void spi_i2s_interrupt_enable(uint32_t spi_periph, uint8_t interrupt);
<b>Function descriptions</b>	enable SPI and I2S interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>interrupt</b>	SPI/I2S interrupt
<i>SPI_I2S_INT_RP</i>	RP interrupt
<i>SPI_I2S_INT_TP</i>	TP interrupt
<i>SPI_I2S_INT_DP</i>	DP interrupt
<i>SPI_I2S_INT_ESTC</i>	end of transfer or suspend or TxFIFO clear interrupt
<i>SPI_I2S_INT_TXF</i>	transmission filled interrupt
<i>SPI_I2S_INT_TXURE</i>	underrun error interrupt
<i>SPI_I2S_INT_RXORE</i>	overrun error interrupt
<i>SPI_I2S_INT_CRCER</i>	CRC error interrupt
<i>SPI_INT_FE</i>	TI frame error interrupt
<i>SPI_I2S_INT_CONF</i>	mode error interrupt
<i>SPI_I2S_INT_TXSERF</i>	TXSER reload interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI0 crc error interrupt */
spi_i2s_interrupt_enable(SPI0, SPI_I2S_INT_CRCER);
```

### spi\_i2s\_interrupt\_disable

The description of spi\_i2s\_interrupt\_disable is shown as below:

Table 3-1088. Function spi\_i2s\_interrupt\_disable

Function name	spi_i2s_interrupt_disable
Function prototype	void spi_i2s_interrupt_disable(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	disable SPI and I2S interrupt
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5
Input parameter{in}	
interrupt	SPI/I2S interrupt
SPI_I2S_INT_RP	RP interrupt
SPI_I2S_INT_TP	TP interrupt
SPI_I2S_INT_DP	DP interrupt
SPI_I2S_INT_ESTC	end of transfer or suspend or TxFIFO clear interrupt
SPI_I2S_INT_TXF	transmission filled interrupt
SPI_I2S_INT_TXURE	underrun error interrupt
SPI_I2S_INT_RXORE	overrun error interrupt
SPI_I2S_INT_CRCER	CRC error interrupt
SPI_INT_FE	Tl frame error interrupt
SPI_I2S_INT_CONFE	mode error interrupt
SPI_I2S_INT_TXSERF	TXSER reload interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable SPI0 crc error interrupt */
```

```
spi_i2s_interrupt_disable(SPI0, SPI_I2S_INT_CRCER);
```

### spi\_i2s\_interrupt\_flag\_get

The description of spi\_i2s\_interrupt\_flag\_get is shown as below:

Table 3-1089. Function spi\_i2s\_interrupt\_flag\_get

Function name	spi_i2s_interrupt_flag_get
Function prototype	FlagStatus spi_i2s_interrupt_flag_get(uint32_t spi_periph, uint8_t interrupt);
Function descriptions	get SPI and I2S interrupt flag status
Precondition	-
The called functions	-
Input parameter{in}	
spi_periph	SPI peripheral
SPIx	x=0,1...5

Input parameter{in}	
<b>interrupt</b>	SPI/I2S interrupt flag status
<i>SPI_I2S_INT_FLAG_RP</i>	RP interrupt flag
<i>SPI_I2S_INT_FLAG_TP</i>	TP interrupt flag
<i>SPI_I2S_INT_FLAG_DP</i>	DP interrupt flag
<i>SPI_I2S_INT_FLAG_ET</i>	end of transfer or receive interrupt flag
<i>SPI_I2S_INT_FLAG_TXF</i>	transmission filled interrupt flag
<i>SPI_I2S_INT_FLAG_TXURERR</i>	underrun error interrupt flag
<i>SPI_I2S_INT_FLAG_RXORERR</i>	overrun error interrupt flag
<i>SPI_I2S_INT_FLAG_RCERR</i>	CRC error interrupt flag
<i>SPI_I2S_INT_FLAG_TFERR</i>	TI frame error interrupt flag
<i>SPI_I2S_INT_FLAG_ONFERR</i>	mode error interrupt flag
<i>SPI_I2S_INT_FLAG_TXSERF</i>	TXSER reload interrupt flag
<i>SPI_I2S_INT_FLAG_SPD</i>	suspend interrupt flag
<i>SPI_I2S_INT_FLAG_TC</i>	TxFIFO clear interrupt flag
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```

/* get SPI0 RP interrupt status */

if((spi_i2s_flag_get(SPI0, SPI_FLAG_RWNE) | (RESET != spi_i2s_interrupt_flag_get(SPI0,
SPI_I2S_INT_FLAG_RP))){

    RxBuffer1[RxCounter1++] = spi_i2s_data_receive(SPI0);

}

```

### spi\_i2s\_flag\_get

The description of spi\_i2s\_flag\_get is shown as below:

Table 3-1090. Function spi\_i2s\_flag\_get

<b>Function name</b>	spi_i2s_flag_get
<b>Function prototype</b>	FlagStatus spi_i2s_flag_get(uint32_t spi_periph, uint32_t flag);
<b>Function descriptions</b>	get SPI and I2S flag status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>flag</b>	SPI/I2S flag status
<i>SPI_FLAG_RP</i>	SPI RP flag
<i>SPI_FLAG_TP</i>	SPI TP flag
<i>SPI_FLAG_DP</i>	SPI DP flag
<i>SPI_FLAG_ET</i>	end of transfer or receive flag
<i>SPI_FLAG_TXF</i>	SPI transmission filled flag
<i>SPI_FLAG_TXURERR</i>	SPI underrun error flag
<i>SPI_FLAG_RXORERR</i>	SPI overrun error flag
<i>SPI_FLAG_CRCERR</i>	SPI CRC error flag
<i>SPI_FLAG_FERR</i>	SPI TI frame error flag
<i>SPI_FLAG_CONFERR</i>	SPI mode error flag
<i>SPI_FLAG_TXSERF</i>	SPI TXSER reload flag
<i>SPI_FLAG_SPD</i>	SPI suspend flag
<i>SPI_FLAG_TC</i>	SPI TxFIFO clear flag
<i>SPI_FLAG_RWNE</i>	the word of SPI RXFIFO is not empty flag
<i>I2S_FLAG_RP</i>	I2S RP flag
<i>I2S_FLAG_TP</i>	I2S TP flag
<i>I2S_FLAG_DP</i>	I2S DP flag
<i>I2S_FLAG_ET</i>	end of transfer or receive flag
<i>I2S_FLAG_TXF</i>	I2S transmission filled flag
<i>I2S_FLAG_TXURERR</i>	I2S underrun error flag
<i>I2S_FLAG_RXORERR</i>	I2S overrun error flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get SPI0 RWNE flag status */
```

```
if((spi_i2s_flag_get(SPI0, SPI_FLAG_RWNE) | (RESET != spi_i2s_interrupt_flag_get(SPI0, SPI_I2S_INT_FLAG_RP))){
```

```
    RxBuffer1[RxCounter1++] = spi_i2s_data_receive(SPI0);
```

}

### spi\_i2s\_flag\_clear

The description of spi\_i2s\_flag\_clear is shown as below:

**Table 3-1091. Function spi\_i2s\_flag\_clear**

<b>Function name</b>	spi_i2s_flag_clear
<b>Function prototype</b>	void spi_i2s_flag_clear(uint32_t spi_periph, uint32_t flag);
<b>Function descriptions</b>	clear SPI and I2S flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>flag</b>	SPI/I2S flag status
<i>SPI_STATC_ETC</i>	clear the end of transfer flag
<i>SPI_STATC_TXFC</i>	clear the send transmission filled flag
<i>SPI_STATC_TXURER RC</i>	clear the transmission underrun error flag
<i>SPI_STATC_RXORER RC</i>	clear the reception overrun error flag
<i>SPI_STATC_CRCERR C</i>	clear the CRC error flag
<i>SPI_STATC_FERRC</i>	clear the SPI TI format error flag
<i>SPI_STATC_CONFER RC</i>	clear the configuration error flag
<i>SPI_STATC_TXSERFC</i>	clear the TXSERF flag
<i>SPI_STATC_SPDC</i>	clear the suspend flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear SPI0 CRC error flag status */
spi_i2s_flag_clear(SPI0, SPI_STATC_CRCERRC);
```

### spi\_i2s\_rxfifo\_plevel\_get

The description of spi\_i2s\_rxfifo\_plevel\_get is shown as below:

**Table 3-1092. Function spi\_i2s\_rxfifo\_plevel\_get**

<b>Function name</b>	spi_i2s_rxfifo_plevel_get
----------------------	---------------------------

<b>Function prototype</b>	uint32_t spi_i2s_rxfifo_plevel_get(uint32_t spi_periph);
<b>Function descriptions</b>	get SPI and I2S RXFIFO packing level
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	2-bit RXFIFO packing data number

Example:

```
/* get SPI0 Rx FIFO packing data frame number */
```

```
uint32_t rxfifo_val;
```

```
rxfifo_val = spi_i2s_rxfifo_plevel_get (SPI0);
```

### spi\_i2s\_remain\_data\_num\_get

The description of spi\_i2s\_remain\_data\_num\_get is shown as below:

**Table 3-1093. Function spi\_i2s\_remain\_data\_num\_get**

<b>Function name</b>	spi_i2s_remain_data_num_get
<b>Function prototype</b>	uint32_t spi_i2s_remain_data_num_get(uint32_t spi_periph);
<b>Function descriptions</b>	get SPI and I2S remaining data frames number in the TXSIZE session
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<b>SPIx</b>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	32-bit SPI and I2S remaining data frames number (0-0xFFFF)

Example:

```
/* get SPI0 TXSIZE value */
```

```
uint32_t txsize_val;
```

```
txsize_val = spi_i2s_remain_data_num_get(SPI0);
```

### spi\_fifo\_threshold\_level\_set

The description of spi\_fifo\_threshold\_level\_set is shown as below:

**Table 3-1094. Function spi\_fifo\_threshold\_level\_set**

<b>Function name</b>	spi_fifo_threshold_level_set
<b>Function prototype</b>	void spi_fifo_threshold_level_set(uint32_t spi_periph, uint32_t fifo_thl);
<b>Function descriptions</b>	set SPI FIFO threshold level
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Input parameter{in}</b>	
<b>fifo_thl</b>	FIFO threshold
<i>SPI_FIFO_TH_xDATA</i>	The data frame number in single data packedt (x = 01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set SPI0 4-byte fifo threshold */
```

```
spi_fifo_threshold_level_set(SPI0, SPI_FIFO_TH_04DATA);
```

### spi\_word\_access\_enable

The description of spi\_word\_access\_enable is shown as below:

**Table 3-1095. Function spi\_word\_access\_enable**

<b>Function name</b>	spi_word_access_enable
<b>Function prototype</b>	void spi_word_access_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI word access
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI0 word access */
```

```
spi_word_access_enable(SPI0);
```



## spi\_word\_access\_disable

The description of spi\_word\_access\_disable is shown as below:

**Table 3-1096. Function spi\_word\_access\_disable**

<b>Function name</b>	spi_word_access_disable
<b>Function prototype</b>	void spi_word_access_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI word access
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 word access */
spi_word_access_disable(SPI0);
```

## spi\_byte\_access\_enable

The description of spi\_byte\_access\_enable is shown as below:

**Table 3-1097. Function spi\_byte\_access\_enable**

<b>Function name</b>	spi_byte_access_enable
<b>Function prototype</b>	void spi_byte_access_enable(uint32_t spi_periph);
<b>Function descriptions</b>	enable SPI byte access
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable SPI0 byte access */
spi_byte_access_enable(SPI0);
```

## spi\_byte\_access\_disable

The description of spi\_byte\_access\_disable is shown as below:

**Table 3-1098. Function spi\_byte\_access\_disable**

<b>Function name</b>	spi_byte_access_disable
<b>Function prototype</b>	void spi_byte_access_disable(uint32_t spi_periph);
<b>Function descriptions</b>	disable SPI byte access
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>spi_periph</b>	SPI peripheral
<i>SPIx</i>	x=0,1...5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable SPI0 byte access */
spi_byte_access_disable(SPI0);
```

## 3.31. SYSCFG

The SYSCFG registers are listed in chapter [3.31.1](#), and the SYSCFG firmware functions are introduced in chapter [3.31.2](#).

### 3.31.1. Descriptions of Peripheral registers

SYSCFG registers are listed in the table shown as below:

**Table 3-1099. SYSCFG Registers**

Registers	Descriptions
SYSCFG_PMCFG	Peripheral mode configuration register
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_LKCTL	Lockup control register
SYSCFG_CPSCCTL	I/O compensation control register
SYSCFG_CPSCCCFG	I/O compensation cell code configuration register
SYSCFG_TIMERCISEL0	Timer input selection register 0
SYSCFG_TIMERCISEL1	Timer input selection register 1

Registers	Descriptions
SYSCFG_TIMERCISEL2	Timer input selection register 2
SYSCFG_TIMERCISEL3	Timer input selection register 3
SYSCFG_TIMERCISEL4	Timer input selection register 4
SYSCFG_TIMERCISEL5	Timer input selection register 5
SYSCFG_TIMERCISEL6	Timer input selection register 6
SYSCFG_CPUICAC	CPU ICACHE error status register
SYSCFG_CPUDCAC	CPU DCACHE error status register
SYSCFG_FPUINTEN	FPU interrupt enable register
SYSCFG_SRAMCFG0	SRAM configuration register 0
SYSCFG_SRAMCFG1	SRAM configuration register 1
SYSCFG_USERCFG	User configuration register

### 3.31.2. Descriptions of Peripheral functions

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

**Table 3-1100. SYSCFG firmware function**

Function name	Function description
syscfg_deinit	reset the SYSCFG registers
syscfg_i2c_fast_mode_plus_enable	enable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
syscfg_i2c_fast_mode_plus_disable	disable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
syscfg_analog_switch_enable	open analog switch
syscfg_analog_switch_disable	close analog switch
syscfg_enet_phy_interface_config	configure the PHY interface for the Ethernet MAC
syscfg_exti_line_config	configure the GPIO pin as EXTI Line
syscfg_lockup_enable	enable module lockup function
syscfg_timer_input_source_select	select timer channel input source
syscfg_compensation_config	configure the I/O compensation cell
syscfg_io_low_voltage_speed_optimization_enable	enable I/O speed optimization, high-speed at low-voltage
syscfg_io_low_voltage_speed_optimization_disable	disable I/O speed optimization, high-speed at low-voltage
syscfg_pnmos_compensation_code_set	set P/N MOS compensation value
syscfg_secure_sram_size_set	set secure SRAM size
syscfg_secure_sram_size_get	get secure SRAM size
syscfg_bootmode_get	get BOOT mode
syscfg_tcm_wait_state_enable	enable TCM wait state
syscfg_tcm_wait_state_disable	disable TCM wait state
syscfg_fpu_interrupt_enable	FPU interrupt enable
syscfg_fpu_interrupt_disable	FPU interrupt disable

Function name	Function description
syscfg_compensation_flag_get	get compensation cell flags
syscfg_cpu_cache_status_get	get ICACHE or DCACHE status
syscfg_brownout_reset_threshold_level_get	get brownout reset threshold level

### Enum timer\_channel\_input\_enum

**Table 3-1101. Enum timer\_channel\_input\_enum**

enum name	Function description
TIMER7_CIO_INPUT_TIMER7_CH0	select CMP1 output as TIMER7 CIO
TIMER7_CIO_INPUT_TIMER7_CH0_MP1_OUT	select TIMER7 CH0 as TIMER7 CIO
TIMER7_CIO_INPUT_TIMER7_CH1	select TIMER7 CH1 as TIMER7 CIO
TIMER7_CIO_INPUT_TIMER7_CH2	select TIMER7 CH2 as TIMER7 CIO
TIMER7_CIO_INPUT_TIMER7_CH3	select TIMER7 CH3 as TIMER7 CIO
TIMER0_CIO_INPUT_TIMER0_CH0	select CMP0 output as TIMER0 CIO
TIMER0_CIO_INPUT_TIMER0_CH0_MP0_OUT	select TIMER0 CH0 as TIMER0 CIO
TIMER0_CIO_INPUT_TIMER0_CH1	select TIMER0 CH1 as TIMER0 CIO
TIMER0_CIO_INPUT_TIMER0_CH2	select TIMER0 CH2 as TIMER0 CIO
TIMER0_CIO_INPUT_TIMER0_CH3	select TIMER0 CH3 as TIMER0 CIO
TIMER2_CIO_INPUT_TIMER2_CH0	select TIMER2 CH0 as TIMER2 CIO
TIMER2_CIO_INPUT_TIMER2_CH0_MP0_OUT	select CMP0 as TIMER2 CIO
TIMER2_CIO_INPUT_TIMER2_CH0_MP1_OUT	select CMP1 as TIMER2 CIO
TIMER2_CIO_INPUT_TIMER2_CH0_MP0_OR_CMP1_OUT	select CMP0 or CMP1 as TIMER2 CIO
TIMER2_CIO_INPUT_TIMER2_CH1	select TIMER2 CH1 as TIMER2 CIO
TIMER2_CIO_INPUT_TIMER2_CH2	select TIMER2 CH2 as TIMER2 CIO
TIMER2_CIO_INPUT_TIMER2_CH3	select TIMER2 CH3 as TIMER2 CIO

enum name	Function description
TIMER1_CIO_INPUT_T IMER1_CH0	select TIMER1 CH0 as TIMER1 CI0
TIMER1_CI1_INPUT_T IMER1_CH1	select TIMER1 CH1 as TIMER1 CI1
TIMER1_CI2_INPUT_T IMER1_CH2	select TIMER1 CH2 as TIMER1 CI2
TIMER1_CI3_INPUT_T IMER1_CH3	select TIMER1 CH3 as TIMER1 CI3
TIMER1_CI3_INPUT_C MP0_OUT	select CMP0 output as TIMER1 CI3
TIMER1_CI3_INPUT_C MP1_OUT	select CMP1 output as TIMER1 CI3
TIMER1_CI3_INPUT_C MP0_OR_CMP1_OUT	select CMP0 or CMP1 output as TIMER1 CI3
TIMER4_CIO_INPUT_T IMER4_CH0	select TIMER4 CH0 as TIMER4 CI0
TIMER4_CI1_INPUT_T IMER4_CH1	select TIMER4 CH1 as TIMER4 CI1
TIMER4_CI2_INPUT_T IMER4_CH2	select TIMER4 CH2 as TIMER4 CI2
TIMER4_CI3_INPUT_T IMER4_CH3	select TIMER4 CH3 as TIMER4 CI3
TIMER3_CIO_INPUT_T IMER3_CH0	select TIMER3 CH0 as TIMER3 CI0
TIMER3_CI1_INPUT_T IMER3_CH1	select TIMER3 CH1 as TIMER3 CI1
TIMER3_CI2_INPUT_T IMER3_CH2	select TIMER3 CH2 as TIMER3 CI2
TIMER3_CI3_INPUT_T IMER3_CH3	select TIMER3 CH3 as TIMER3 CI3
TIMER23_CIO_INPUT_ TIMER23_CH0	select TIMER23 CH0 as TIMER23 CI0
TIMER23_CI1_INPUT_ TIMER23_CH1	select TIMER23 CH1 as TIMER23 CI1
TIMER23_CI2_INPUT_ TIMER23_CH2	select TIMER23 CH2 as TIMER23 CI2
TIMER23_CI3_INPUT_ TIMER23_CH3	select TIMER23 CH3 as TIMER23 CI3
TIMER22_CIO_INPUT_ TIMER22_CH0	select TIMER22 CH0 as TIMER22 CI0
TIMER22_CI1_INPUT_ TIMER22_CH1	select TIMER22 CH1 as TIMER22 CI1

enum name	Function description
TIMER22_CI2_INPUT_ TIMER22_CH2	select TIMER22 CH2 as TIMER22 CI2
TIMER22_CI3_INPUT_ TIMER22_CH3	select TIMER22 CH3 as TIMER22 CI3
TIMER22_CI3_INPUT_ CMP0_OUT	select CMP0 output as TIMER22 CI3
TIMER22_CI3_INPUT_ CMP1_OUT	select CMP1 output as TIMER22 CI3
TIMER22_CI3_INPUT_ CMP0_OR_CMP1_OU T	select CMP0 or CMP1 output as TIMER22 CI3
TIMER14_CI0_INPUT_ TIMER14_CH0	select TIMER14 CH0 as TIMER14 CI0
TIMER14_CI0_INPUT_ TIMER1_CH0	select TIMER1 CH0 as TIMER14 CI0
TIMER14_CI0_INPUT_ TIMER2_CH0	select TIMER2 CH0 as TIMER14 CI0
TIMER14_CI0_INPUT_ TIMER3_CH0	select TIMER3 CH0 as TIMER14 CI0
TIMER14_CI0_INPUT_ LXTAL	select LXTAL as TIMER14 CI0
TIMER14_CI0_INPUT_ LPIRC4M	select LPIRC4M as TIMER14 CI0
TIMER14_CI0_INPUT_ CKOUT1	select CKOUT1 as TIMER14 CI0
TIMER14_CI1_INPUT_ TIMER14_CH1	select TIMER14 CH1 as TIMER14 CI1
TIMER14_CI1_INPUT_ TIMER1_CH1	select TIMER1 CH1 as TIMER14 CI1
TIMER14_CI1_INPUT_ TIMER2_CH1	select TIMER2 CH1 as TIMER14 CI1
TIMER14_CI1_INPUT_ TIMER3_CH1	select TIMER3 CH1 as TIMER14 CI1
TIMER40_CI0_INPUT_ TIMER40_CH0	select TIMER40 CH0 as TIMER40 CI0
TIMER40_CI0_INPUT_ TIMER2_CH0	select TIMER2 CH0 as TIMER40 CI0
TIMER40_CI0_INPUT_ TIMER3_CH0	select TIMER3 CH0 as TIMER40 CI0
TIMER40_CI0_INPUT_ TIMER4_CH0	select TIMER4 CH0 as TIMER40 CI0
TIMER40_CI0_INPUT_ LXTAL	select LXTAL as TIMER40 CI0

enum name	Function description
LXTAL	
TIMER40_CIO_INPUT_ LPIRC4M	select LPIRC4M as TIMER40 CIO
TIMER40_CIO_INPUT_ CKOUT1	select CKOUT1 as TIMER40 CIO
TIMER40_C11_INPUT_ TIMER40_CH1	select TIMER40 CH1 as TIMER40 CIO
TIMER40_C11_INPUT_ TIMER2_CH1	select TIMER2 CH1 as TIMER40 CIO
TIMER40_C11_INPUT_ TIMER3_CH1	select TIMER3 CH1 as TIMER40 CIO
TIMER40_C11_INPUT_ TIMER4_CH1	select TIMER4 CH1 as TIMER40 CIO
TIMER41_CIO_INPUT_ TIMER41_CH0	select TIMER41 CH0 as TIMER41 CIO
TIMER41_CIO_INPUT_ TIMER3_CH0	select TIMER3 CH0 as TIMER41 CIO
TIMER41_CIO_INPUT_ TIMER4_CH0	select TIMER4 CH0 as TIMER41 CIO
TIMER41_CIO_INPUT_ TIMER22_CH0	select TIMER22 CH0 as TIMER41 CIO
TIMER41_CIO_INPUT_ LXTAL	select LXTAL as TIMER41 CIO
TIMER41_CIO_INPUT_ LPIRC4M	select LPIRC4M as TIMER41 CIO
TIMER41_CIO_INPUT_ CKOUT1	select CKOUT1 as TIMER41 CIO
TIMER41_C11_INPUT_ TIMER41_CH1	select TIMER41 CH1 as TIMER41 C11
TIMER41_C11_INPUT_ TIMER3_CH1	select TIMER3 CH1 as TIMER41 C11
TIMER41_C11_INPUT_ TIMER4_CH1	select TIMER4 CH1 as TIMER41 C11
TIMER41_C11_INPUT_ TIMER22_CH1	select TIMER22 CH1 as TIMER41 C11
TIMER42_CIO_INPUT_ TIMER42_CH0	select TIMER42 CH0 as TIMER42 CIO
TIMER42_CIO_INPUT_ TIMER4_CH0	select TIMER4 CH0 as TIMER42 CIO
TIMER42_CIO_INPUT_ TIMER22_CH0	select TIMER22 CH0 as TIMER42 CIO
TIMER42_CIO_INPUT_ TIMER23_CH0	select TIMER23 CH0 as TIMER42 CIO

enum name	Function description
TIMER23_CH0	
TIMER42_CIO_INPUT_ LXTAL	select LXTAL as TIMER42 CIO
TIMER42_CIO_INPUT_ LPIRC4M	select LPIRC4M as TIMER42 CIO
TIMER42_CIO_INPUT_ CKOUT1	select CKOUT1 as TIMER42 CIO
TIMER42_C11_INPUT_ TIMER42_CH1	select TIMER42 CH1 as TIMER42 C11
TIMER42_C11_INPUT_ TIMER4_CH1	select TIMER4 CH1 as TIMER42 C11
TIMER42_C11_INPUT_ TIMER22_CH1	select TIMER22 CH1 as TIMER42 C11
TIMER42_C11_INPUT_ TIMER23_CH1	select TIMER23 CH1 as TIMER42 C11
TIMER15_CIO_INPUT_ TIMER15_CH0	select TIMER15 CH0 as TIMER15 CIO
TIMER15_CIO_INPUT_ RC32K	select IRC32K as TIMER15 CIO
TIMER15_CIO_INPUT_ LXTAL	select LXTAL as TIMER15 CIO
TIMER15_CIO_INPUT_ WKUP_IT	select WKUP IT as TIMER15 CIO
TIMER16_CIO_INPUT_ TIMER16_CH0	select TIMER16 CH0 as TIMER16 CIO
TIMER16_CIO_INPUT_ HXTAL_RTCDIV	select HXTAL/RTCDIV 1M as TIMER16 CIO
TIMER16_CIO_INPUT_ CKOUT0	select CKOUT0 as TIMER16 CIO
TIMER43_CIO_INPUT_ TIMER43_CH0	select TIMER43 CH0 as TIMER43 CIO
TIMER43_CIO_INPUT_ TIMER22_CH0	select TIMER22 CH0 as TIMER43 CIO
TIMER43_CIO_INPUT_ TIMER23_CH0	select TIMER23 CH0 as TIMER43 CIO
TIMER43_CIO_INPUT_ LXTAL	select LXTAL as TIMER43 CIO
TIMER43_CIO_INPUT_ LPIRC4M	select LPIRC4M as TIMER43 CIO
TIMER43_CIO_INPUT_ CKOUT1	select CKOUT1 as TIMER43 CIO
TIMER43_C11_INPUT_ TIMER43_CH1	select TIMER43 CH1 as TIMER43 C11



enum name	Function description
TIMER43_CH1	
TIMER43_CI1_INPUT_ TIMER22_CH1	select TIMER22 CH1 as TIMER43 CI1
TIMER43_CI1_INPUT_ TIMER23_CH1	select TIMER23 CH1 as TIMER43 CI1
TIMER44_CI0_INPUT_ TIMER44_CH0	select TIMER44 CH0 as TIMER44 CI0
TIMER44_CI0_INPUT_ TIMER23_CH0	select TIMER23 CH0 as TIMER44 CI0
TIMER44_CI0_INPUT_ LXTAL	select LXTAL as TIMER44 CI0
TIMER44_CI0_INPUT_ LPIRC4M	select LPIRC4M as TIMER44 CI0
TIMER44_CI0_INPUT_ CKOUT1	select CKOUT1 as TIMER44 CI0
TIMER44_CI1_INPUT_ TIMER44_CH1	select TIMER44 CH1 as TIMER44 CI1
TIMER44_CI1_INPUT_ TIMER23_CH1	select TIMER23 CH1 as TIMER44 CI1

## syscfg\_deinit

The description of syscfg\_deinit is shown as below:

**Table 3-1102. Function syscfg\_deinit**

<b>Function name</b>	syscfg_deinit
<b>Function prototype</b>	void syscfg_deinit(void);
<b>Function descriptions</b>	reset the SYSCFG 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 SYSCFG registers */
syscfg_deinit();
```

### syscfg\_i2c\_fast\_mode\_plus\_enable

The description of syscfg\_i2c\_fast\_mode\_plus\_enable is shown as below:

**Table 3-1103. Function syscfg\_i2c\_fast\_mode\_plus\_enable**

<b>Function name</b>	syscfg_i2c_fast_mode_plus_enable
<b>Function prototype</b>	void syscfg_i2c_fast_mode_plus_enable(uint32_t i2c_fmp);
<b>Function descriptions</b>	enable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>i2c_fmp</b>	I2C fast mode plus function
<i>SYSCFG_I2C0_FMP</i>	I2C0 fast mode plus
<i>SYSCFG_I2C1_FMP</i>	I2C1 fast mode plus
<i>SYSCFG_I2C2_FMP</i>	I2C2 fast mode plus
<i>SYSCFG_I2C3_FMP</i>	I2C3 fast mode plus
<i>SYSCFG_I2C_FMP_P</i> <i>B6</i>	I2C fast mode plus on PB6 pin
<i>SYSCFG_I2C_FMP_P</i> <i>B7</i>	I2C fast mode plus on PB7 pin
<i>SYSCFG_I2C_FMP_P</i> <i>B8</i>	I2C fast mode plus on PB8 pin
<i>SYSCFG_I2C_FMP_P</i> <i>B9</i>	I2C fast mode plus on PB9 pin
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I2C0 fast mode plus function */
syscfg_i2c_fast_mode_plus_enable(SYSCFG_I2C0_FMP);
```

### syscfg\_i2c\_fast\_mode\_plus\_disable

The description of syscfg\_i2c\_fast\_mode\_plus\_disable is shown as below:

**Table 3-1104. Function syscfg\_i2c\_fast\_mode\_plus\_disable**

<b>Function name</b>	syscfg_i2c_fast_mode_plus_disable
<b>Function prototype</b>	void syscfg_i2c_fast_mode_plus_disable(uint32_t i2c_fmp);
<b>Function descriptions</b>	disable I2Cx(x=0,1,2,3) fast mode plus or I2C fast mode plus PBx(x=6,7,8,9)
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>i2c_fmp</b>	I2C fast mode plus function
<i>SYSCFG_I2C0_FMP</i>	I2C0 fast mode plus
<i>SYSCFG_I2C1_FMP</i>	I2C1 fast mode plus
<i>SYSCFG_I2C2_FMP</i>	I2C2 fast mode plus
<i>SYSCFG_I2C3_FMP</i>	I2C3 fast mode plus
<i>SYSCFG_I2C_FMP_P</i> <i>B6</i>	I2C fast mode plus on PB6 pin
<i>SYSCFG_I2C_FMP_P</i> <i>B7</i>	I2C fast mode plus on PB7 pin
<i>SYSCFG_I2C_FMP_P</i> <i>B8</i>	I2C fast mode plus on PB8 pin
<i>SYSCFG_I2C_FMP_P</i> <i>B9</i>	I2C fast mode plus on PB9 pin
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable I2C0 fast mode plus function */
```

```
syscfg_i2c_fast_mode_plus_disable(SYSCFG_I2C0_FMP);
```

### syscfg\_analog\_switch\_enable

The description of syscfg\_analog\_switch\_enable is shown as below:

**Table 3-1105. Function syscfg\_analog\_switch\_enable**

<b>Function name</b>	syscfg_analog_switch_enable
<b>Function prototype</b>	void syscfg_analog_switch_enable(uint32_t gpio_answ);
<b>Function descriptions</b>	open analog switch
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>gpio_answ</b>	GPIO analog switch
<i>SYSCFG_PA0_ANALOG_SWITCH</i>	PA0 analog switch
<i>SYSCFG_PA1_ANALOG_SWITCH</i>	PA1 analog switch
<i>SYSCFG_PC2_ANALOG_SWITCH</i>	PC2 analog switch
<i>SYSCFG_PC3_ANALOG_SWITCH</i>	PC3 analog switch
Output parameter{out}	

-	-
Return value	
-	-

Example:

```
/* open PA0 analog switch function */
```

```
syscfg_analog_switch_enable(SYSCFG_PA0_ANALOG_SWITCH);
```

### syscfg\_analog\_switch\_disable

The description of syscfg\_analog\_switch\_disable is shown as below:

**Table 3-1106. Function syscfg\_analog\_switch\_disable**

<b>Function name</b>	syscfg_analog_switch_disable
<b>Function prototype</b>	void syscfg_analog_switch_disable(uint32_t gpio_answ);
<b>Function descriptions</b>	close analog switch
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>gpio_answ</b>	GPIO analog switch
SYSCFG_PA0_ANALOG_SWITCH	PA0 analog switch
SYSCFG_PA1_ANALOG_SWITCH	PA1 analog switch
SYSCFG_PC2_ANALOG_SWITCH	PC2 analog switch
SYSCFG_PC3_ANALOG_SWITCH	PC3 analog switch
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* close PA0 analog switch function */
```

```
syscfg_analog_switch_disable(SYSCFG_PA0_ANALOG_SWITCH);
```

### syscfg\_enet\_phy\_interface\_config

The description of syscfg\_enet\_phy\_interface\_config is shown as below:

**Table 3-1107. Function syscfg\_enet\_phy\_interface\_config**

<b>Function name</b>	syscfg_enet_phy_interface_config
<b>Function prototype</b>	void syscfg_enet_phy_interface_config(uint32_t ethernet, uint32_t

	phy_interface);
<b>Function descriptions</b>	configure the PHY interface for the ethernet MAC
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>ethernet</b>	Ethernet
<i>ENET0</i>	Ethernet 0
<i>ENET1</i>	Ethernet 1
<b>Input parameter{in}</b>	
<b>phy_interface</b>	specifies the media interface mode
<i>SYSCFG_ENET_PHY_MII</i>	MII mode is selected
<i>SYSCFG_ENET_PHY_RMII</i>	RMII mode is selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the PHY interface for the Ethernet 0 MAC */
```

```
syscfg_enet_phy_interface_config(ENET0, SYSCFG_ENET_PHY_MII);
```

### syscfg\_exti\_line\_config

The description of syscfg\_exti\_line\_config is shown as below:

**Table 3-1108. Function syscfg\_exti\_line\_config**

<b>Function name</b>	syscfg_exti_line_config
<b>Function prototype</b>	void syscfg_exti_line_config(uint8_t exti_port, uint8_t exti_pin);
<b>Function descriptions</b>	configure the GPIO pin as EXTI Line
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>exti_port</b>	specify the GPIO port used in EXTI
<i>EXTI_SOURCE_GPIOx</i>	x = A,B,C,D,E,F,G,H,J,K
<b>Input parameter{in}</b>	
<b>exti_pin</b>	specify the EXTI line
<i>EXTI_SOURCE_PINx</i>	GPIOA x = 0..15, GPIOB x = 0..15, GPIOC x = 0..15, GPIOD x = 0..15, GPIOE x = 0..15, GPIOF x = 0..15, GPIOG x = 0..15, GPIOH x = 0..15, GPIOI x = 0..15, GPIOJ x = 8,9,10,11, GPIOK x = 0,1,2,4,5,6
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* configure the PA0 pin as EXTI Line */
```

```
syscfg_exti_line_config(EXTI_SOURCE_GPIOA, EXTI_SOURCE_PIN0);
```

### syscfg\_lockup\_enable

The description of syscfg\_lockup\_enable is shown as below:

**Table 3-1109. 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>	enable module lockup function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lockup</b>	lockup function
<i>SYSCFG_LVD_LOCKUP</i>	LVD signal
<i>SYSCFG_CPU_LOCKUP</i>	CPU lockup signal
<i>SYSCFG_BKPRAM_LOCKUP</i>	Region 2 backup SRAM ECC double error signal
<i>SYSCFG_SRAM1_LOCKUP</i>	Region 1 SRAM1 ECC double error signal
<i>SYSCFG_SRAM0_LOCKUP</i>	Region 1 SRAM0 ECC double error signal
<i>SYSCFG_DTCM_LOCKUP</i>	Region 0 DTCM ECC double error signal
<i>SYSCFG_ITCM_LOCKUP</i>	Region 0 ITCM-RAM ECC double error signal
<i>SYSCFG_AXIRAM_LOCKUP</i>	Region 0 AXI-SRAM ECC double error signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable module lockup function */
```

```
syscfg_lockup_enable(SYSCFG_CPU_LOCKUP);
```

## syscfg\_lockup\_disable

The description of syscfg\_lockup\_disable is shown as below:

**Table 3-1110. Function syscfg\_lockup\_disable**

<b>Function name</b>	syscfg_lockup_disable
<b>Function prototype</b>	void syscfg_lockup_disable(uint32_t lockup);
<b>Function descriptions</b>	disable module lockup function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>lockup</b>	lockup function
SYSCFG_LVD_LOCKUP	LVD signal
SYSCFG_CPU_LOCKUP	CPU lockup signal
SYSCFG_BKPRAM_LOCKUP	Region 2 backup SRAM ECC double error signal
SYSCFG_SRAM1_LOCKUP	Region 1 SRAM1 ECC double error signal
SYSCFG_SRAM0_LOCKUP	Region 1 SRAM0 ECC double error signal
SYSCFG_DTCM_LOCKUP	Region 0 DTCM ECC double error signal
SYSCFG_ITCM_LOCKUP	Region 0 ITCM-RAM ECC double error signal
SYSCFG_AXIRAM_LOCKUP	Region 0 AXI-SRAM ECC double error signal
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable module lockup function */
syscfg_lockup_disable(SYSCFG_CPU_LOCKUP);
```

## syscfg\_timer\_input\_source\_select

The description of syscfg\_timer\_input\_source\_select is shown as below:

**Table 3-1111. 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>	
<b>timer_input</b>	TIMER channel input select, refer to <a href="#">Table 3-1101. Enum timer_channel_input_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\_compensation\_config

The description of syscfg\_compensation\_config is shown as below:

**Table 3-1112. Function syscfg\_compensation\_config**

<b>Function name</b>	syscfg_compensation_config
<b>Function prototype</b>	void syscfg_compensation_config(uint32_t syscfg_compensation);
<b>Function descriptions</b>	configure the I/O compensation cell
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>syscfg_compensation</b>	syscfg compensation
SYSCFG_COMPENSATION_ENABLE	I/O compensation cell is enabled
SYSCFG_COMPENSATION_DISABLE	I/O compensation cell is disabled
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the I/O compensation cell */
syscfg_compensation_config(SYSCFG_COMPENSATION_ENABLE);
```

### syscfg\_io\_low\_voltage\_speed\_optimization\_enable

The description of syscfg\_io\_low\_voltage\_speed\_optimization\_enable is shown as below:



**Table 3-1113. Function syscfg\_io\_low\_voltage\_speed\_optimization\_enable**

<b>Function name</b>	syscfg_io_low_voltage_speed_optimization_enable
<b>Function prototype</b>	void syscfg_io_low_voltage_speed_optimization_enable(void);
<b>Function descriptions</b>	enable I/O speed optimization, high-speed at low-voltage
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable I/O speed optimization, high-speed at low-voltage */
syscfg_io_low_voltage_speed_optimization_enable();
```

### syscfg\_io\_low\_voltage\_speed\_optimization\_disable

The description of syscfg\_io\_low\_voltage\_speed\_optimization\_disable is shown as below:

**Table 3-1114. Function syscfg\_io\_low\_voltage\_speed\_optimization\_disable**

<b>Function name</b>	syscfg_io_low_voltage_speed_optimization_disable
<b>Function prototype</b>	void syscfg_io_low_voltage_speed_optimization_disable(void);
<b>Function descriptions</b>	disable I/O speed optimization, high-speed at low-voltage
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable I/O speed optimization, high-speed at low-voltage */
syscfg_io_low_voltage_speed_optimization_disable();
```

### syscfg\_pnmos\_compensation\_code\_set

The description of syscfg\_pnmos\_compensation\_code\_set is shown as below:

**Table 3-1115. Function syscfg\_pnmos\_compensation\_code\_set**

<b>Function name</b>	syscfg_pnmos_compensation_code_set
----------------------	------------------------------------

<b>Function prototype</b>	void syscfg_pmos_compensation_code_set(uint32_t mos, uint32_t code);
<b>Function descriptions</b>	set P/N MOS compensation value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mos</b>	P/N MOS
<i>NMOS_COMPENSATION</i>	NMOS
<i>PMOS_COMPENSATION</i>	PMOS
<b>Input parameter{in}</b>	
<b>code</b>	P/N MOS compensation value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set PMOS compensation value */
```

```
syscfg_pmos_compensation_code_set(PMOS_COMPENSATION, 0x02);
```

### syscfg\_secure\_sram\_size\_set

The description of syscfg\_secure\_sram\_size\_set is shown as below:

**Table 3-1116. Function syscfg\_secure\_sram\_size\_set**

<b>Function name</b>	syscfg_secure_sram_size_set
<b>Function prototype</b>	void syscfg_secure_sram_size_set(uint32_t size);
<b>Function descriptions</b>	set secure SRAM size
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>size</b>	secure SRAM size
<i>SECURE_SRAM_SIZE_0KB</i>	secure SRAM size is 0KB
<i>SECURE_SRAM_SIZE_32KB</i>	secure SRAM size is 32KB
<i>SECURE_SRAM_SIZE_64KB</i>	secure SRAM size is 64KB
<i>SECURE_SRAM_SIZE_128KB</i>	secure SRAM size is 128KB
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* set secure SRAM size */

syscfg_secure_sram_size_set(SECURE_SRAM_SIZE_32KB);
```

### syscfg\_secure\_sram\_size\_get

The description of syscfg\_secure\_sram\_size\_get is shown as below:

**Table 3-1117. Function syscfg\_secure\_sram\_size\_get**

<b>Function name</b>	syscfg_secure_sram_size_get
<b>Function prototype</b>	uint32_t syscfg_secure_sram_size_get(void);
<b>Function descriptions</b>	get secure SRAM size
<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>	SRAM size
SECURE_SRAM_SIZE_0KB	secure SRAM size is 0KB
SECURE_SRAM_SIZE_32KB	secure SRAM size is 32KB
SECURE_SRAM_SIZE_64KB	secure SRAM size is 64KB
SECURE_SRAM_SIZE_128KB	secure SRAM size is 128KB

Example:

```
/* get secure SRAM size */

uint32_t ret_val = 0U;

ret_val = syscfg_secure_sram_size_get();
```

### syscfg\_bootmode\_get

The description of syscfg\_bootmode\_get is shown as below:

**Table 3-1118. Function syscfg\_bootmode\_get**

<b>Function name</b>	syscfg_bootmode_get
<b>Function prototype</b>	uint32_t syscfg_bootmode_get(void);
<b>Function descriptions</b>	get BOOT mode

<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>	BOOT mode
<i>BOOT_SRAM</i>	BOOT from SRAM (ITCM/DTCM/RAM shared/AXI SRAM)
<i>BOOT_SECURITY</i>	BOOT from Security
<i>BOOT_SYSTEM</i>	BOOT_SYS (BootLoader)
<i>BOOT_USER_FLASH</i>	BOOT_USER (User flash OSPI0/1)

Example:

```
/* get BOOT mode */

uint32_t boot_mod = 0U;

boot_mod = syscfg_bootmode_get();
```

### syscfg\_tcm\_wait\_state\_enable

The description of syscfg\_tcm\_wait\_state\_enable is shown as below:

**Table 3-1119. Function syscfg\_tcm\_wait\_state\_enable**

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

Example:

```
/* enable TCM wait state */

syscfg_tcm_wait_state_enable();
```

### syscfg\_tcm\_wait\_state\_disable

The description of syscfg\_tcm\_wait\_state\_disable is shown as below:

**Table 3-1120. Function syscfg\_tcm\_wait\_state\_disable**

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

Example:

```
/* disable TCM wait state */
```

```
syscfg_tcm_wait_state_disable();
```

### syscfg\_fpu\_interrupt\_enable

The description of syscfg\_fpu\_interrupt\_enable is shown as below:

**Table 3-1121. Function syscfg\_fpu\_interrupt\_enable**

<b>Function name</b>	syscfg_fpu_interrupt_enable
<b>Function prototype</b>	void syscfg_fpu_interrupt_enable(uint32_t fpu_int);
<b>Function descriptions</b>	enable FPU interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fpu_int</b>	FPU interrupt
SYSCFG_FPUINT_INEXACT	inexact interrupt
SYSCFG_FPUINT_INPUT_ABNORMAL	input abnormal interrupt
SYSCFG_FPUINT_OVERFLOW	overflow interrupt
SYSCFG_FPUINT_UNDERFLOW	underflow interrupt
SYSCFG_FPUINT_DIV0	divide-by-zero interrupt
SYSCFG_FPUINT_INVALID_OPERATION	invalid operation interrupt
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* enable FPU inexact interrupt */
syscfg_fpu_interrupt_enable(SYSCFG_FPUINT_INEXACT);
```

### syscfg\_fpu\_interrupt\_disable

The description of syscfg\_fpu\_interrupt\_disable is shown as below:

**Table 3-1122. Function syscfg\_fpu\_interrupt\_disable**

Function name	syscfg_fpu_interrupt_disable
Function prototype	void syscfg_fpu_interrupt_disable(uint32_t fpu_int);
Function descriptions	disable FPU interrupt
Precondition	-
The called functions	-
Input parameter{in}	
fpu_int	FPU interrupt
SYSCFG_FPUINT_INEXACT	inexact interrupt
SYSCFG_FPUINT_INPUT_ABNORMAL	input abnormal interrupt
SYSCFG_FPUINT_OVERFLOW	overflow interrupt
SYSCFG_FPUINT_UNDERFLOW	underflow interrupt
SYSCFG_FPUINT_DIV0	divide-by-zero interrupt
SYSCFG_FPUINT_INVALID_OPERATION	invalid operation interrupt
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable FPU inexact interrupt */
syscfg_fpu_interrupt_disable(SYSCFG_FPUINT_INEXACT);
```

### syscfg\_compensation\_flag\_get

The description of syscfg\_compensation\_flag\_get is shown as below:

Table 3-1123. Function syscfg\_compensation\_flag\_get

<b>Function name</b>	syscfg_compensation_flag_get
<b>Function prototype</b>	FlagStatus syscfg_compensation_flag_get(uint32_t cps_flag);
<b>Function descriptions</b>	get compensation cell flags
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cps_flag</b>	compensation cell flag
<i>SYSCFG_FLAG_IO_LOW_VOLTAGE</i>	I/O in low voltage state flag, product supply voltage is working below 2.5V
<i>SYSCFG_FLAG_COMPENSATION_READY</i>	I/O compensation cell ready flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get compensation cell flags */
```

```
FlagStatus flag;
```

```
flag = syscfg_compensation_flag_get(SYSCFG_FLAG_IO_LOW_VOLTAGE);
```

### syscfg\_cpu\_cache\_status\_get

The description of syscfg\_cpu\_cache\_status\_get is shown as below:

Table 3-1124. Function syscfg\_cpu\_cache\_status\_get

<b>Function name</b>	syscfg_cpu_cache_status_get
<b>Function prototype</b>	uint32_t syscfg_cpu_cache_status_get(uint32_t cache, uint32_t status);
<b>Function descriptions</b>	get the ICACHE or DCACHE detection and error information
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>cache</b>	cache
<i>ICACHE_STATUS</i>	ICACHE status
<i>DCACHE_STATUS</i>	DCACHE status
<b>Input parameter{in}</b>	
<b>status</b>	status
<i>CPU_CACHE_ERROR_DETECTION</i>	select detection information
<i>CPU_CACHE_ERROR_BANK</i>	select error information
<b>Output parameter{out}</b>	

-	-
<b>Return value</b>	
<b>uint32_t</b>	ICACHE or DCACHE value

Example:

```
/* get the ICACHE detection and error information */
```

```
uint32_t cache;
```

```
cache = syscfg_cpu_cache_status_get(ICACHE_STATUS, CPU_CACHE_ERROR_BANK);
```

### **syscfg\_brownout\_reset\_threshold\_level\_get**

The description of syscfg\_brownout\_reset\_threshold\_level\_get is shown as below:

**Table 3-1125. Function syscfg\_brownout\_reset\_threshold\_level\_get**

<b>Function name</b>	syscfg_brownout_reset_threshold_level_get
<b>Function prototype</b>	uint32_t syscfg_brownout_reset_threshold_level_get(void);
<b>Function descriptions</b>	get brownout reset threshold level
<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>	brownout reset threshold level
<i>BOR_OFF</i>	no BOR function
<i>BOR_THRESHOLD_V</i> <i>AL1</i>	BOR threshold value 1
<i>BOR_THRESHOLD_V</i> <i>AL2</i>	BOR threshold value 2
<i>BOR_THRESHOLD_V</i> <i>AL3</i>	BOR threshold value 3

Example:

```
/* get brownout reset threshold level */
```

```
uint32_t val;
```

```
val = syscfg_brownout_reset_threshold_level_get();
```

## **3.32. TIMER**

The timers have a counter that can be used as an unsigned counter and supports both input capture and output compare. Timers (TIMERx) are divided into five sorts: advanced timer



(TIMERx, x=0, 7), general level0 timer (TIMERx, x=1~4, 22, 23), general level3 timer (TIMERx, x=14, 40~44), general level4 timer (TIMERx, x=15, 16) and basic timer (TIMERx, x=5, 6, 50, 51). The specific functions of different types of timer are different. The TIMER registers are listed in chapter [3.32.1](#), the TIMER firmware functions are introduced in chapter [3.32.2](#).

### 3.32.1. Descriptions of Peripheral registers

TIMERx registers are listed in the table shown as below:

**Table 3-1126. TIMERx Registers**

Registers	Descriptions
TIMER_CTL0	Control register 0
TIMER_CTL1	Control register 1
TIMER_SMCFG	Slave mode configuration register
TIMER_DMAINTEN	DMA and interrupt enable register
TIMER_INTF	Interrupt flag register
TIMER_SWEVG	Software event generation register
TIMER_CHCTL0	Channel control register 0
TIMER_CHCTL1	Channel control register 1
TIMER_CHCTL2	Channel control register 2
TIMER_CNT	Counter register
TIMER_CNTL	TIMER counter low register (only for TIMERx, x=50,51)
TIMER_PSC	Prescaler register
TIMER_CAR	Counter auto reload register
TIMER_CARL	TIMER counter auto reload low register (only for TIMERx, x=50,51)
TIMER_CREP0	Counter repetition register 0
TIMER_CH0CV	Channel 0 capture/compare value register
TIMER_CH1CV	Channel 1 capture/compare value register
TIMER_CH2CV	Channel 2 capture/compare value register
TIMER_CH3CV	Channel 3 capture/compare value register
TIMER_CCHP	Channel complementary protection register
TIMER_MCHCTL0	TIMER multi mode channel control register 0
TIMER_MCHCTL1	TIMER multi mode channel control register 1
TIMER_MCHCTL2	TIMER multi mode channel control register 2
TIMER_MCH0CV	TIMER multi mode channel 0 capture or compare value register
TIMER_MCH1CV	TIMER multi mode channel 1 capture or compare value register
TIMER_MCH2CV	TIMER multi mode channel 2 capture or compare value register
TIMER_MCH3CV	TIMER multi mode channel 3 capture or compare value register
TIMER_CH0COMV_ADD	TIMER channel 0 additional compare value register
TIMER_CH1COMV_ADD	TIMER channel 1 additional compare value register
TIMER_CH2COMV_ADD	TIMER channel 2 additional compare value register

Registers	Descriptions
TIMER_CH3COMV_ADD	TIMER channel 3 additional compare value register
TIMER_CTL2	TIMER control register 2
TIMER_FCCHP0	TIMER free complementary channel protection register 0
TIMER_FCCHP1	TIMER free complementary channel protection register 1
TIMER_FCCHP2	TIMER free complementary channel protection register 2
TIMER_FCCHP3	TIMER free complementary channel protection register 3
TIMER_AFCTL0	TIMER alternate function control register 0
TIMER_AFCTL1	TIMER alternate function control register 1
TIMER_WDGPWR	TIMER watchdog counter period register
TIMER_CREP1	TIMER counter repetition register 1
TIMER_CNTH	TIMER counter high register (only for TIMEx, x=50,51)
TIMER_CARH	TIMER counter auto reload high register (only for TIMEx, x=50,51)
TIMER_DMACFG	DMA configuration register
TIMER_DMATB	DMA transfer buffer register
TIMER_CFG	Configuration register

### 3.32.2. Descriptions of Peripheral functions

The description format of firmware functions are shown as below:

**Table 3-1127. TIMEx firmware function**

Function name	Function description
timer_deinit	deinit a TIMER
timer_struct_para_init	initialize the parameters of TIMER init parameter struct with the default values
timer_init	initialize TIMER counter
timer_enable	enable a timer
timer_disable	disable a timer
timer_auto_reload_shadow_enable	enable the auto reload shadow function
timer_auto_reload_shadow_disable	disable the auto reload shadow function
timer_update_event_enable	enable the update event
timer_update_event_disable	disable the update event
timer_counter_alignment	set TIMER counter alignment mode
timer_counter_up_direction	set TIMER counter up direction
timer_counter_down_direction	set TIMER counter down direction
timer_prescaler_config	configure TIMER prescaler
timer_repetition_value_config	configure TIMER repetition register value
timer_runtime_repetition_value_read	configure TIMER runtime repetition register value
timer_autoreload_value_config	configure TIMER autoreload register value
timer_autoreload_value_read	read TIMER autoreload register value
timer_counter_value_config	configure TIMER counter register value
timer_counter_read	read TIMER counter value

Function name	Function description
timer_prescaler_read	read TIMER prescaler value
timer_single_pulse_mode_config	configure TIMER single pulse mode
timer_delayable_single_pulse_mode_config	configure timer delayable single pulse mode
timer_update_source_config	configure TIMER update source
timer_dma_enable	enable the TIMER DMA
timer_dma_disable	disable the TIMER DMA
timer_channel_dma_request_source_select	channel DMA request source selection
timer_dma_transfer_config	configure the TIMER DMA transfer
timer_event_software_generate	software generate events
timer_break_struct_para_init	initialize the parameters of TIMER break parameter struct with the default values
timer_break_config	configure TIMER break function
timer_break_enable	enable TIMER break function
timer_break_disable	disable TIMER break function
timer_automatic_output_enable	enable TIMER output automatic function
timer_automatic_output_disable	disable TIMER output automatic function
timer_primary_output_config	configure TIMER primary output function
timer_channel_control_shadow_config	configure channel commutation control shadow register
timer_channel_control_shadow_update_config	configure TIMER channel control shadow register update control
timer_channel_output_struct_para_init	initialize the parameters of TIMER channel output parameter struct with the default values
timer_channel_output_config	configure TIMER channel output function
timer_channel_output_mode_config	configure TIMER channel output compare mode
timer_channel_output_pulse_value_config	configure TIMER channel output pulse value
timer_channel_output_shadow_config	configure TIMER channel output shadow function
timer_channel_output_clear_config	configure TIMER channel output clear function
timer_channel_output_polarity_config	configure TIMER channel output polarity
timer_channel_complementary_output_polarity_config	configure TIMER channel complementary output polarity
timer_channel_output_state_config	configure TIMER channel enable state
timer_channel_complementary_output_state_config	configure TIMER channel complementary output enable state
timer_channel_input_struct_para_init	initialize the parameters of TIMER channel input parameter struct with the default values
timer_input_capture_config	configure TIMER input capture parameter

Function name	Function description
timer_channel_input_capture_prescaler_config	configure TIMER channel input capture prescaler value
timer_channel_capture_value_register_read	read TIMER channel capture compare register value
timer_input_pwm_capture_config	configure TIMER input pwm capture function
timer_hall_mode_config	configure TIMER hall sensor mode
timer_multi_mode_channel_output_parameter_struct_init	initialize TIMER multi mode channel output parameter struct
timer_multi_mode_channel_output_config	configure TIMER multi mode channel output function
timer_multi_mode_channel_mode_config	multi mode channel mode select
timer_input_trigger_source_select	select TIMER input trigger source
timer_master_output0_trigger_source_select	select TIMER master mode output 0 trigger source
timer_master_output1_trigger_source_select	select TIMER master mode output 1 trigger source
timer_slave_mode_select	select TIMER slave mode
timer_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_non_quadrature_decoder_mode_config	configure TIMER non-quadrature decoder mode
timer_internal_clock_config	configure TIMER internal clock mode
timer_internal_trigger_as_external_clock_config	configure TIMER the internal trigger as external clock input
timer_external_trigger_as_external_clock_config	configure TIMER the external trigger as external clock input
timer_external_clock_mode0_config	configure TIMER the external clock mode 0
timer_external_clock_mode1_config	configure TIMER the external clock mode 1
timer_external_clock_mode1_disable	disable TIMER the external clock mode 1
timer_write_chxval_register_config	configure TIMER write CHxVAL register selection
timer_output_value_selection_config	configure TIMER output value selection
timer_commutation_control_shadow_register_config	configure commutation control shadow register update selection
timer_output_match_pulse_select	configure TIMER output match pulse selection
timer_channel_composite_pwm_mode_config	configure the TIMER composite PWM mode
timer_channel_composite_pwm_mode_output_pulse	configure the TIMER composite PWM mode output pulse

Function name	Function description
de_output_pulse_value_config	value
timer_channel_additional_compare_value_config	configure TIMER channel additional compare value
timer_channel_additional_output_shadow_config	configure TIMER channel additional output shadow function
timer_channel_additional_compare_value_read	read TIMER channel additional compare value
timer_break_external_source_config	configure TIMER break external source
timer_break_external_polarity_config	configure TIMER break polarity
timer_break_lock_config	configure TIMER break lock function
timer_break_lock_release_config	configure the TIMER break lock release function
timer_channel_break_control_config	configure the TIMER channel break function
timer_channel_dead_time_config	configure the TIMER channel dead time function
timer_free_complementary_struct_parameter_init	initialize TIMER channel free complementary parameter struct with a default value
timer_channel_free_complementary_config	configure channel free complementary protection
timer_watchdog_value_config	configure quadrature decoder signal disconnection detection watchdog value
timer_watchdog_value_read	read quadrature decoder signal disconnection detection watchdog value
timer_decoder_disconnection_detection_config	configure quadrature decoder signal disconnection detection function
timer_decoder_jump_detection_config	configure decoder signal jump detection function
timer_upif_backup_config	configure the UPIF bit backup function
timer_upifbu_bit_get	get the UPIFBU bit in the TIMEx_CNT register
timer_flag_get	get TIMER flags
timer_flag_clear	clear TIMER flags
timer_interrupt_enable	enable the TIMER interrupt
timer_interrupt_disable	disable the TIMER interrupt
timer_interrupt_flag_get	get timer interrupt flags
timer_interrupt_flag_clear	clear TIMER interrupt flags

## Structure timer\_parameter\_struct

Table 3-1128. 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,

Member name	Function description
	TIMER_COUNTER_CENTER_BOTH)
counterdirection	counter direction(TIMER_COUNTER_UP, TIMER_COUNTER_DOWN)
period	period value(0~0xFFFF(TIMERx(x=0,7,14~16,40~44)), 0~0xFFFFFFFF(TIMERx(x=1~6,22,23)), 0~0xFFFFFFFFFFFFFFFF(TIMERx(x=50,51)))
clockdivision	clock division value(TIMER_CKDIV_DIV1, TIMER_CKDIV_DIV2, TIMER_CKDIV_DIV4)
repetitioncounter	the counter repetition value(0~0xFF, use TIMER_CREP0 register; 0xFF~0xFFFFFFFF, use TIMER_CREP1 register)

### Structure timer\_break\_parameter\_struct

**Table 3-1129. Structure timer\_break\_parameter\_struct**

Member name	Function description
runoffstate	run mode off-state(TIMER_ROS_STATE_ENABLE, TIMER_ROS_STATE_DISABLE)
ideloffstate	idle mode off-state(TIMER_IOS_STATE_ENABLE, TIMER_IOS_STATE_DISABLE)
deadtime	dead time(0~255)
outputautostate	output automatic enable (TIMER_OUTAUTO_ENABLE, TIMER_OUTAUTO_DISABLE)
protectmode	complementary register protect control(TIMER_CCHP_PROT_OFF, TIMER_CCHP_PROT_0, TIMER_CCHP_PROT_1, TIMER_CCHP_PROT_2)
break0state	BREAK0 input enable (TIMER_BREAK0_ENABLE, TIMER_BREAK0_DISABLE)
break0filter	BREAK0 input filter(0~15)
break0polarity	BREAK0 input polarity(TIMER_BREAK0_POLARITY_LOW, TIMER_BREAK0_POLARITY_HIGH)
break0lock	BREAK0 input lock(TIMER_BREAK0_LK_ENABLE, TIMER_BREAK0_LK_DISABLE)
break0release	BREAK0 input release(TIMER_BREAK0_RELEASE, TIMER_BREAK0_UNRELEASE)
break1state	BREAK1 input enable (TIMER_BREAK1_ENABLE, TIMER_BREAK1_DISABLE)
break1filter	BREAK1 input filter(0~15)
break1polarity	BREAK1 input polarity(TIMER_BREAK1_POLARITY_LOW, TIMER_BREAK1_POLARITY_HIGH)
break1lock	BREAK1 input lock(TIMER_BREAK1_LK_ENABLE, TIMER_BREAK1_LK_DISABLE)
break1release	BREAK1 input release(TIMER_BREAK1_RELEASE, TIMER_BREAK1_UNRELEASE)

## Structure timer\_oc\_parameter\_struct

**Table 3-1130. 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 3-1131. 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 3-1132. Structure timer\_ic\_parameter\_struct**

Member name	Function description
icpolarity	channel input polarity(TIMER_IC_POLARITY_RISING, TIMER_IC_POLARITY_FALLING, TIMER_IC_POLARITY_BOTH_EDGE)
icselection	channel input mode selection(TIMER_IC_SELECTION_DIRECTTI, TIMER_IC_SELECTION_INDIRECTTI, TIMER_IC_SELECTION_ITS, TIMER_IC_SELECTION_PAIR)
icprescaler	channel input capture prescaler(TIMER_IC_PSC_DIV1, TIMER_IC_PSC_DIV2, TIMER_IC_PSC_DIV4, TIMER_IC_PSC_DIV8)
icfilter	channel input capture filter control(0~15)

## Structure timer\_free\_complementary\_parameter\_struct

**Table 3-1133. Structure timer\_free\_complementary\_parameter\_struct**

Member name	Function description
freecomstate	free complementary channel protection enable(TIMER_FCCHP_STATE_ENABLE, TIMER_FCCHP_STATE_DISABLE)
runoffstate	run mode off-state(TIMER_ROS_STATE_ENABLE, TIMER_ROS_STATE_DISABLE)
ideloffstate	idle mode off-state(TIMER_IOS_STATE_ENABLE, TIMER_IOS_STATE_DISABLE)
deadtime	dead time(0~255)

## timer\_deinit

The description of timer\_deinit is shown as below:

**Table 3-1134. 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
TIMERx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)	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 3-1135. 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



Precondition	-
The called functions	-
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to <a href="#">Structure timer_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER init parameter struct with a default value */
```

```
timer_parameter_struct timer_initpara;
```

```
timer_struct_para_init(&timer_initpara);
```

## timer\_init

The description of timer\_init is shown as below:

**Table 3-1136. Function timer\_init**

Function name	timer_init
Function prototype	void timer_init(uint32_t timer_periph, timer_parameter_struct* initpara);
Function descriptions	initialize TIMER counter
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)	TIMER peripheral selection
Input parameter{in}	
initpara	TIMER init parameter struct, the structure members can refer to <a href="#">Structure timer_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER0 */
```

```
timer_parameter_struct timer_initpara;
```

```
timer_initpara.prescaler = 107;
```

```
timer_initpara.alignedmode = TIMER_COUNTER_EDGE;
```

```

timer_initpara.counterdirection = TIMER_COUNTER_UP;

timer_initpara.period          = 999;

timer_initpara.clockdivision   = TIMER_CKDIV_DIV1;

timer_initpara.repetitioncounter = 1;

timer_init(TIMER0,&timer_initpara);

```

### timer\_enable

The description of timer\_enable is shown as below:

**Table 3-1137. Function timer\_enable**

<b>Function name</b>	timer_enable
<b>Function prototype</b>	void timer_enable(uint32_t timer_periph);
<b>Function descriptions</b>	enable a TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* enable TIMER0 */

timer_enable(TIMER0);

```

### timer\_disable

The description of timer\_disable is shown as below:

**Table 3-1138. Function timer\_disable**

<b>Function name</b>	timer_disable
<b>Function prototype</b>	void timer_disable(uint32_t timer_periph);
<b>Function descriptions</b>	disable a TIMER
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)</i>	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 3-1139. Function timer\_auto\_reload\_shadow\_enable**

Function name	timer_auto_reload_shadow_enable
Function prototype	void timer_auto_reload_shadow_enable(uint32_t timer_periph);
Function descriptions	enable the auto reload shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TIMER0 auto reload shadow function */
timer_auto_reload_shadow_enable(TIMER0);
```

### timer\_auto\_reload\_shadow\_disable

The description of timer\_auto\_reload\_shadow\_disable is shown as below:

**Table 3-1140. Function timer\_auto\_reload\_shadow\_disable**

Function name	timer_auto_reload_shadow_disable
Function prototype	void timer_auto_reload_shadow_disable (uint32_t timer_periph);
Function descriptions	disable the auto reload shadow function
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral

<i>TIMERx</i> ( <i>x</i> =0~7,14~16,2 2,23,40~44,50,51)	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 3-1141. 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</i> ( <i>x</i> =0~7,14~16,2 2,23,40~44,50,51)	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 3-1142. 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>	-

Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~7, 14~16, 22, 23, 40~44, 50, 51)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable TIMER0 the update event */
```

```
timer_update_event_disable (TIMER0);
```

### timer\_counter\_alignment

The description of timer\_counter\_alignment is shown as below:

**Table 3-1143. 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>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4, 7, 22, 23)	TIMER peripheral selection
Input parameter{in}	
<b>aligned</b>	alignment mode
<i>TIMER_COUNTER_EDGE</i>	edge-aligned mode
<i>TIMER_COUNTER_CENTRAL_DOWN</i>	center-aligned and counting down assert mode
<i>TIMER_COUNTER_CENTRAL_UP</i>	center-aligned and counting up assert mode
<i>TIMER_COUNTER_CENTRAL_BOTH</i>	center-aligned and counting up/down assert mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set TIMER0 counter center-aligned and counting up assert mode */
```

```
timer_counter_alignment(TIMERO, TIMER_COUNTER_CENTER_UP);
```

### timer\_counter\_up\_direction

The description of timer\_counter\_up\_direction is shown as below:

**Table 3-1144. 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 TIMEx 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>TIMEx(x=0~4,7,22,23)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set TIMERO counter up direction */
timer_counter_up_direction(TIMERO);
```

### timer\_counter\_down\_direction

The description of timer\_counter\_down\_direction is shown as below:

**Table 3-1145. 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 TIMEx 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>TIMEx(x=0~4,7,22,23)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set TIMERO counter down direction */
```

```
timer_counter_down_direction(TIMER0);
```

### timer\_prescaler\_config

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

**Table 3-1146. 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, 14~16, 22, 23, 40~44, 50, 51)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>prescaler</b>	prescaler value (0~0xFFFF)
<b>Input parameter{in}</b>	
<b>pscreload</b>	prescaler reload mode
<i>TIMER_PSC_RELOAD_NOW</i>	the prescaler is loaded right now
<i>TIMER_PSC_RELOAD_UPDATE</i>	the prescaler is loaded at the next update event
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 prescaler */
```

```
timer_prescaler_config(TIMER0, 3000, TIMER_PSC_RELOAD_NOW);
```

### timer\_repetition\_value\_config

The description of timer\_repetition\_value\_config is shown as below:

**Table 3-1147. Function timer\_repetition\_value\_config**

<b>Function name</b>	timer_repetition_value_config
<b>Function prototype</b>	void timer_repetition_value_config(uint32_t timer_periph, uint16_t ccsl, uint32_t repetition);
<b>Function descriptions</b>	configure TIMER repetition register value
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
<b>ccsel</b>	repetition register selection
<i>TIMER_CREP0_ENABLE</i>	the update event rate is depended to <i>TIMERx_CREP0</i> register
<i>TIMER_CREP1_ENABLE</i>	the update event rate is depended to <i>TIMERx_CREP1</i> register
Input parameter{in}	
<b>repetition</b>	the counter repetition value (0~0xFF, use <i>TIMER_CREP0</i> register; 0~0xFFFFFFFF, use <i>TIMER_CREP1</i> register)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 repetition register 0 value */
timer_repetition_value_config(TIMER0, TIMER_CREP0_ENABLE, 98);
```

### timer\_runtime\_repetition\_value\_read

The description of `timer_runtime_repetition_value_read` is shown as below:

**Table 3-1148. Function `timer_runtime_repetition_value_read`**

<b>Function name</b>	<code>timer_runtime_repetition_value_read</code>
<b>Function prototype</b>	<code>uint32_t timer_runtime_repetition_value_read(uint32_t timer_periph);</code>
<b>Function descriptions</b>	read TIMER runtime repetition register value
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
Output parameter{out}	
-	-
Return value	
<b>uint32_t</b>	counter repetition value in <i>TIMER_CREP1</i> register (0~0xFFFFFFFF)

Example:

```
/* read TIMER0 runtime repetition register value */
```



```
uint32_t i = 0;
```

```
i = timer_runtime_repetition_value_read(TIMER0);
```

### timer\_autoreload\_value\_config

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

**Table 3-1149. Function timer\_autoreload\_value\_config**

<b>Function name</b>	timer_autoreload_value_config
<b>Function prototype</b>	void timer_autoreload_value_config(uint32_t timer_periph, uint64_t autoreload);
<b>Function descriptions</b>	configure TIMER autoreload register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~7,14~16,22,23,40~44,50,51)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>autoreload</b>	the counter auto-reload value 0~0xFFFF:TIMERx(x=0,2,3,7,14~16,40~44) 0~0xFFFFFFFF, TIMERx(x=1,4,5,6,22,23) 0~0xFFFFFFFFFFFFFFFF, TIMERx(x=50,51)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER autoreload register value */
```

```
timer_autoreload_value_config(TIMER0, 3000);
```

### timer\_autoreload\_value\_read

The description of timer\_autoreload\_value\_read is shown as below:

**Table 3-1150. Function timer\_autoreload\_value\_read**

<b>Function name</b>	timer_autoreload_value_read
<b>Function prototype</b>	uint64_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</i> ( <i>x</i> =0~7,14~16,22,23,40~44,50,51)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint64_t</b>	counter auto reload register value 0~0xFFFF, <i>TIMERx</i> ( <i>x</i> =0,2,3,7,14~16,40~44) 0~0xFFFFFFFF, <i>TIMERx</i> ( <i>x</i> =1,4,5,6,22,23) 0~0xFFFFFFFFFFFFFFFF, <i>TIMERx</i> ( <i>x</i> =50,51)

Example:

```
/* get TIMER autoreload register value */
```

```
uint64_t i = 0;
```

```
i =(uint64_t) timer_autoreload_value_read (TIMER0);
```

### timer\_counter\_value\_config

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

**Table 3-1151. 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, uint64_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</i> ( <i>x</i> =0~7,14~16,22,23,40~44,50,51)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>counter</b>	the counter value 0~0xFFFF, <i>TIMERx</i> ( <i>x</i> =0,2,3,7,14~16,40~44) 0~0xFFFFFFFF, <i>TIMERx</i> ( <i>x</i> =1,4,5,6,22,23) 0~0xFFFFFFFFFFFFFFFF, <i>TIMERx</i> ( <i>x</i> =50,51)
<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 3-1152. Function timer\_counter\_read**

<b>Function name</b>	timer_counter_read
<b>Function prototype</b>	uint64_t timer_counter_read(uint32_t timer_periph);
<b>Function descriptions</b>	read TIMER counter value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,40~44,50,51)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint64_t</b>	counter value 0~0xFFFF, TIMERx(x=0,2,3,7,14~16,40~44) 0~0xFFFFFFFF, TIMERx(x=1,4,5,6,22,23) 0~0xFFFFFFFFFFFFFFFF, TIMERx(x=50,51)

Example:

```
/* read TIMER0 counter value */
uint64_t i = 0;
i = timer_counter_read(TIMER0);
```

## timer\_prescaler\_read

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

**Table 3-1153. 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>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,40~44,50,51)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

<b>uint16_t</b>	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 3-1154. 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=0~7, 14~16, 22, 23, 40~44, 50, 51)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>spmode</b>	pulse mode
<i>TIMER_SP_MODE_SINGLE</i>	single pulse mode
<i>TIMER_SP_MODE_REPETITIVE</i>	repetitive pulse mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 single pulse mode */
```

```
timer_single_pulse_mode_config(TIMER0, TIMER_SP_MODE_SINGLE);
```

### timer\_delayable\_single\_pulse\_mode\_config

The description of timer\_delayable\_single\_pulse\_mode\_config is shown as below:

**Table 3-1155. Function timer\_delayable\_single\_pulse\_mode\_config**

<b>Function name</b>	timer_delayable_single_pulse_mode_config
<b>Function prototype</b>	void timer_delayable_single_pulse_mode_config(uint32_t timer_periph,

	uint16_t channel, uint32_t dspmode, uint16_t cnt_dir);
<b>Function descriptions</b>	configure timer delayable single pulse mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14,40~44)
<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>dspmode</b>	delayable SPM mode
<i>TIMER_OC_MODE_DS</i> <i>PM0</i>	delayable SPM mode 0
<i>TIMER_OC_MODE_DS</i> <i>PM1</i>	delayable SPM mode1
<b>Input parameter{in}</b>	
<b>cnt_dir</b>	counter direction selection
<i>TIMER_COUNTER_UP</i>	count up
<i>TIMER_COUNTER_DO</i> <i>WN</i>	count down
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0_CH0 delayable single pulse mode */
```

```
timer_delayable_single_pulse_mode_config(TIMER0,                TIMER_CH_0,
TIMER_OC_MODE_DSPM0, TIMER_COUNTER_UP);
```

### timer\_update\_source\_config

The description of timer\_update\_source\_config is shown as below:

Table 3-1156. Function timer\_update\_source\_config

Function name	timer_update_source_config
Function prototype	void timer_update_source_config(uint32_t timer_periph, uint32_t update);
Function descriptions	configure TIMER update source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)	TIMER peripheral selection
Input parameter{in}	
update	update source
TIMER_UPDATE_SRC_GLOBAL	any of the following events generate an update interrupt or DMA request: <ul style="list-style-type: none"> <li>– The UPG bit is set</li> <li>– The counter generates an overflow or underflow event</li> <li>– The slave mode controller generates an update event</li> </ul>
TIMER_UPDATE_SRC_REGULAR	only counter overflow/underflow generates an update interrupt or DMA request.
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER update only by counter overflow/underflow */
```

```
timer_update_source_config(TIMER0, TIMER_UPDATE_SRC_REGULAR);
```

### timer\_dma\_enable

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

Table 3-1157. Function timer\_dma\_enable

Function name	timer_dma_enable
Function prototype	void timer_dma_enable(uint32_t timer_periph, uint32_t dma);
Function descriptions	enable the TIMER DMA
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)	please refer to the following parameters
Input parameter{in}	
dma	timer DMA source enable
TIMER_DMA_UPD	update DMA request, TIMERx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)

<i>TIMER_DMA_CH0D</i>	channel 0 DMA request, TIMERx (x=0~4,7,14~16,22,23,40~44)
<i>TIMER_DMA_CH1D</i>	channel 1 DMA request, TIMERx (x=0~4,7,14,22,23,40~44)
<i>TIMER_DMA_CH2D</i>	channel 2 DMA request, TIMERx (x=0~4,7,22,23)
<i>TIMER_DMA_CH3D</i>	channel 3 DMA request, TIMERx (x=0~4,7,22,23)
<i>TIMER_DMA_CMTD</i>	commutation DMA request, TIMERx (x=0,7,14~16,40~44)
<i>TIMER_DMA_TRGD</i>	trigger DMA request, TIMERx (x=0~4,7,14,22,23,40~44)
<i>TIMER_DMA_MCH0D</i>	multi mode channel 0 DMA request, TIMERx (x=0,7,14~16,40~44)
<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 3-1158. Function timer\_dma\_disable**

<b>Function name</b>	timer_dma_disable
<b>Function prototype</b>	void timer_dma_disable (uint32_t timer_periph, uint32_t dma);
<b>Function descriptions</b>	disable the TIMER DMA
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> (x=0~7,14~16,22,23,40~44,50,51)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>dma</b>	timer DMA source disable
<i>TIMER_DMA_UPD</i>	update DMA request, TIMERx(x=0~7,14~16,22,23,40~44,50,51)
<i>TIMER_DMA_CH0D</i>	channel 0 DMA request, TIMERx (x=0~4,7,14~16,22,23,40~44)
<i>TIMER_DMA_CH1D</i>	channel 1 DMA request, TIMERx (x=0~4,7,14,22,23,40~44)
<i>TIMER_DMA_CH2D</i>	channel 2 DMA request, TIMERx (x=0~4,7,22,23)
<i>TIMER_DMA_CH3D</i>	channel 3 DMA request, TIMERx (x=0~4,7,22,23)
<i>TIMER_DMA_CMTD</i>	commutation DMA request, TIMERx (x=0,7,14~16,40~44)
<i>TIMER_DMA_TRGD</i>	trigger DMA request, TIMERx (x=0~4,7,14,22,23,40~44)
<i>TIMER_DMA_MCH0D</i>	multi mode channel 0 DMA request, TIMERx (x=0,7,14~16,40~44)
<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, <i>TIMERx</i> (x=0,7)
<i>TIMER_DMA_MCH3D</i>	multi mode channel 3 DMA request, <i>TIMERx</i> (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 3-1159. Function `timer_channel_dma_request_source_select`**

<b>Function name</b>	<code>timer_channel_dma_request_source_select</code>
<b>Function prototype</b>	<code>void timer_channel_dma_request_source_select(uint32_t timer_periph, uint32_t dma_request);</code>
<b>Function descriptions</b>	channel DMA request source selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> (x=0~4, 7, 14~16, 22, 23, 40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>dma_request</b>	channel DMA request source selection
<i>TIMER_DMAREQUEST_CHANNELEVENT</i>	DMA request of channel n is sent when channel n event occurs
<i>TIMER_DMAREQUEST_UPDATEEVENT</i>	DMA request of channel n is sent when update event occurs
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* TIMER0 channel DMA request of channel n is sent when channel event occurs */
```

```
timer_channel_dma_request_source_select(TIMER0,
TIMER_DMAREQUEST_CHANNELEVENT);
```



## timer\_dma\_transfer\_config

The description of timer\_dma\_transfer\_config is shown as below:

**Table 3-1160. 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
TIMERx(x=0~4,7,14~16,22,23,40~44)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>dma_baseaddr</b>	DMA transfer access start address
TIMER_DMACFG_DMA TA_CTL0	DMA transfer address is TIMER_CTL0, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_CTL1	DMA transfer address is TIMER_CTL1, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_SMCFG	DMA transfer address is TIMER_SMCFG, TIMERx(x=0~4,7,14,22,23,40~44)
TIMER_DMACFG_DMA TA_DMAINTEN	DMA transfer address is TIMER_DMAINTEN, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_INTF	DMA transfer address is TIMER_INTF, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_SWEVG	DMA transfer address is TIMER_SWEVG, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_CHCTL0	DMA transfer address is TIMER_CHCTL0, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_CHCTL1	DMA transfer address is TIMER_CHCTL1, TIMERx(x=0~4,7,22,23)
TIMER_DMACFG_DMA TA_CHCTL2	DMA transfer address is TIMER_CHCTL2, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_CNT	DMA transfer address is TIMER_CNT, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_PSC	DMA transfer address is TIMER_PSC, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_CAR	DMA transfer address is TIMER_CAR, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_DMACFG_DMA TA_CREP0	DMA transfer address is TIMER_CREP0, TIMERx(x=0,7,14~16,40~44)
TIMER_DMACFG_DMA	DMA transfer address is TIMER_CH0CV,

<i>TA_CH0CV</i>	TIMERx(x=0~4,7,14~16,22,23,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH1CV</i>	DMA transfer address is TIMER_CH1CV, TIMERx(x=0~4,7,14,22,23)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH2CV</i>	DMA transfer address is TIMER_CH2CV, TIMERx(x=0~4,7,22,23)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH3CV</i>	DMA transfer address is TIMER_CH3CV, TIMERx(x=0~4,7,22,23)
<i>TIMER_DMACFG_DMA</i> <i>TA_CCHP</i>	DMA transfer address is TIMER_CCHP, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL0</i>	DMA transfer address is TIMER_MCHCTL0, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL1</i>	DMA transfer address is TIMER_MCHCTL1, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCHCTL2</i>	DMA transfer address is TIMER_MCHCTL2, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCH0CV</i>	DMA transfer address is TIMER_MCH0CV, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCH1CV</i>	DMA transfer address is TIMER_MCH1CV, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCH2CV</i>	DMA transfer address is TIMER_TIMER_MCH2CV, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_MCH3CV</i>	DMA transfer address is TIMER_TIMER_MCH3CV, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH0COMV_ADD</i>	DMA transfer address is TIMER_CH0COMV_ADD, TIMERx(x=0~4,7,14,22,23)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH1COMV_ADD</i>	DMA transfer address is TIMER_CH1COMV_ADD, TIMERx(x=0~4,7,14,22,23)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH2COMV_ADD</i>	DMA transfer address is TIMER_CH2COMV_ADD, TIMERx(x=0~4,7,22,23)
<i>TIMER_DMACFG_DMA</i> <i>TA_CH3COMV_ADD</i>	DMA transfer address is TIMER_CH3COMV_ADD, TIMERx(x=0~4,7,22,23)
<i>TIMER_DMACFG_DMA</i> <i>TA_CTL2</i>	DMA transfer address is TIMER_CTL2, (x=0~4,7,14~16,22,23,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_FCCHP0</i>	DMA transfer address is TIMER_FCCHP0, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_DMACFG_DMA</i> <i>TA_FCCHP1</i>	DMA transfer address is TIMER_FCCHP1, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_FCCHP2</i>	DMA transfer address is TIMER_FCCHP2, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_FCCHP3</i>	DMA transfer address is TIMER_FCCHP3, TIMERx(x=0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_AFCTL0</i>	DMA transfer address is TIMER_AFCTL0, TIMERx(x=0,7,14~16,40~44)

<i>TIMER_DMACFG_DMA</i> <i>TA_AFCTL1</i>	DMA transfer address is <i>TIMER_AFCTL1</i> , <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_DMACFG_DMA</i> <i>TA_WDGCNT</i>	DMA transfer address is <i>TIMER_WDGCNT</i> , <i>TIMERx</i> ( <i>x</i> =0~4,7,22,23)
<i>TIMER_DMACFG_DMA</i> <i>TA_CREP1</i>	DMA transfer address is <i>TIMER_CREP1</i> , <i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)
<b>Input parameter{in}</b>	
<b>dma_lenth</b>	DMA transfer count
<i>TIMER_DMACFG_DMA</i> <i>TC_xTRANSFER</i>	( <i>x</i> =1~38), 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 3-1161. 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</i> ( <i>x</i> =0~7,14~16,22,23,40~44,50,51)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>event</b>	the timer software event generation sources
<i>TIMER_EVENT_SRC_UPG</i>	update event, <i>TIMERx</i> ( <i>x</i> =0~7,14~16,22,23,40~44,50,51)
<i>TIMER_EVENT_SRC_CH0G</i>	channel 0 capture or compare event generation, <i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,22,23,40~44)
<i>TIMER_EVENT_SRC_CH1G</i>	channel 1 capture or compare event generation, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44)
<i>TIMER_EVENT_SRC_CH2G</i>	channel 2 capture or compare event generation, <i>TIMERx</i> ( <i>x</i> =0~4,7,22,23)

<i>H2G</i>	
<i>TIMER_EVENT_SRC_C</i> <i>H3G</i>	channel 3 capture or compare event generation, $TIMERx(x=0\sim4,7,22,23)$
<i>TIMER_EVENT_SRC_C</i> <i>MTG</i>	channel commutation event generation, $TIMERx(x=0,7,14\sim16,40\sim44)$
<i>TIMER_EVENT_SRC_T</i> <i>RGG</i>	trigger event generation, $TIMERx(x=0\sim4,7,14,22,23,40\sim44)$
<i>TIMER_EVENT_SRC_B</i> <i>RK0G</i>	BREAK0 event generation, $TIMERx(x=0,7,14\sim16,40\sim44)$
<i>TIMER_EVENT_SRC_B</i> <i>RK1G</i>	BREAK1 event generation, $TIMERx(x=0,7)$
<i>TIMER_EVENT_SRC_M</i> <i>CH0G</i>	multi mode channel 0 capture or compare event generation, $TIMERx(x=0,7,14\sim16,40\sim44)$
<i>TIMER_EVENT_SRC_M</i> <i>CH1G</i>	multi mode channel 1 capture or compare event generation, $TIMERx(x=0,7)$
<i>TIMER_EVENT_SRC_M</i> <i>CH2G</i>	multi mode channel 2 capture or compare event generation, $TIMERx(x=0,7)$
<i>TIMER_EVENT_SRC_M</i> <i>CH3G</i>	multi mode channel 3 capture or compare event generation, $TIMERx(x=0,7)$
<i>TIMER_EVENT_SRC_C</i> <i>H0COMADDG</i>	channel 0 additional compare event generation, $TIMERx(x=0\sim4,7,14,22,23)$
<i>TIMER_EVENT_SRC_C</i> <i>H1COMADDG</i>	channel 1 additional compare event generation, $TIMERx(x=0\sim4,7,14,22,23)$
<i>TIMER_EVENT_SRC_C</i> <i>H2COMADDG</i>	channel 2 additional compare event generation, $TIMERx(x=0\sim4,7,22,23)$
<i>TIMER_EVENT_SRC_C</i> <i>H3COMADDG</i>	channel 3 additional compare event generation, $TIMERx(x=0\sim4,7,22,23)$
<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 3-1162. Function timer\_break\_struct\_para\_init**

<b>Function name</b>	timer_break_struct_para_init
<b>Function prototype</b>	void timer_break_struct_para_init(timer_break_parameter_struct* breakpara);

<b>Function descriptions</b>	initialize the parameters of TIMER break parameter struct with the default values
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>breakpara</b>	TIMER break parameter struct, the structure members can refer to <a href="#">Structure timer break parameter struct.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize TIMER break parameter struct with a default value */
```

```
timer_break_parameter_struct timer_breakpara;
```

```
timer_break_struct_para_init(&timer_breakpara);
```

### timer\_break\_config

The description of timer\_break\_config is shown as below:

**Table 3-1163. Function timer\_break\_config**

<b>Function name</b>	timer_break_config
<b>Function prototype</b>	void timer_break_config(uint32_t timer_periph, timer_break_parameter_struct* breakpara);
<b>Function descriptions</b>	configure TIMER break function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0,7,14~16,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>breakpara</b>	TIMER break parameter struct, the structure members can refer to <a href="#">Structure timer break parameter struct.</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
timer_break_parameter_struct timer_breakpara;
```

```
timer_breakpara.runoffstate = TIMER_ROS_STATE_DISABLE;
```

```

timer_breakpara.ideloffstate    = TIMER_IOS_STATE_DISABLE;

timer_breakpara.deadtime        = 0U;

timer_breakpara.outputautostate = TIMER_OUTAUTO_ENABLE;

timer_breakpara.protectmode     = TIMER_CCHP_PROT_OFF;

timer_breakpara.break0state     = TIMER_BREAK0_ENABLE;

timer_breakpara.break0filter    = 0U;

timer_breakpara.break0polarity  = TIMER_BREAK0_POLARITY_LOW;

timer_breakpara.break0bidirectional = TIMER_BREAK0_LK_ENABLE;

timer_breakpara.break0release   = TIMER_BREAK0_UNRELEASE;

timer_breakpara.break1state     = TIMER_BREAK1_DISABLE;

timer_breakpara.break1filter    = 0U;

timer_breakpara.break1polarity  = TIMER_BREAK1_POLARITY_LOW;

timer_breakpara.break1bidirectional = TIMER_BREAK1_LK_DISABLE;

timer_breakpara.break1release   = TIMER_BREAK1_UNRELEASE;

timer_break_config(TIMER0, &timer_breakpara);

```

### timer\_break\_enable

The description of timer\_break\_enable is shown as below:

**Table 3-1164. Function timer\_break\_enable**

Function name	timer_break_enable
Function prototype	void timer_break_enable(uint32_t timer_periph, uint16_t break_num);
Function descriptions	enable TIMER break function
Precondition	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7,14~16,40~44)	TIMER peripheral selection
Input parameter{in}	
break_num	TIMER BREAKx
TIMER_BREAK0	BREAK0 input signals, TIMERx(x=0,7,14~16,40~44)
TIMER_BREAK1	BREAK1 input signals, TIMERx(x=0,7)
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```
/* enable TIMER0 BREAK0 function*/
timer_break_enable(TIMER0, TIMER_BREAK0);
```

### timer\_break\_disable

The description of timer\_break\_disable is shown as below:

**Table 3-1165. Function timer\_break\_disable**

<b>Function name</b>	timer_break_disable
<b>Function prototype</b>	void timer_break_disable(uint32_t timer_periph, uint16_t break_num);
<b>Function descriptions</b>	disable TIMER break function
<b>Precondition</b>	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>break_num</b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	BREAK0 input signals, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_BREAK1</i>	BREAK1 input signals, TIMERx(x=0,7)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable TIMER0 BREAK0 function*/
timer_break_disable(TIMER0, TIMER_BREAK0);
```

### timer\_automatic\_output\_enable

The description of timer\_automatic\_output\_enable is shown as below:

**Table 3-1166. Function timer\_automatic\_output\_enable**

<b>Function name</b>	timer_automatic_output_enable
<b>Function prototype</b>	void timer_automatic_output_enable(uint32_t timer_periph);
<b>Function descriptions</b>	enable TIMER output automatic function
<b>Precondition</b>	This function can be called only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	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 3-1167. 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 <i>TIMERx_CCHP</i> register is 00.
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	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 3-1168. Function timer\_primary\_output\_config**

<b>Function name</b>	timer_primary_output_config
----------------------	-----------------------------



<b>Function prototype</b>	void timer_primary_output_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure TIMER primary output function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TIMER0 primary output function */
```

```
timer_primary_output_config(TIMER0, ENABLE);
```

### timer\_channel\_control\_shadow\_config

The description of timer\_channel\_control\_shadow\_config is shown as below:

**Table 3-1169. Function timer\_channel\_control\_shadow\_config**

<b>Function name</b>	timer_channel_control_shadow_config
<b>Function prototype</b>	void timer_channel_control_shadow_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure channel commutation control shadow register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* channel commutation control shadow register enable */
timer_channel_control_shadow_config(TIMER0, ENABLE);
```

### timer\_channel\_control\_shadow\_update\_config

The description of timer\_channel\_control\_shadow\_update\_config is shown as below:

**Table 3-1170. Function timer\_channel\_control\_shadow\_update\_config**

<b>Function name</b>	timer_channel_control_shadow_update_config
<b>Function prototype</b>	void timer_channel_control_shadow_update_config(uint32_t timer_periph, uint32_t ccuctl);
<b>Function descriptions</b>	configure TIMER channel control shadow register update control
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>ccuctl</b>	channel control shadow register update control
<i>TIMER_UPDATECTL_CCU</i>	the shadow registers update by when CMTG bit is set
<i>TIMER_UPDATECTL_CUTRI</i>	the shadow registers update by when CMTG bit is set or an rising edge of TRGI occurs
<i>TIMER_UPDATECTL_CCUOVER</i>	the shadow registers update by when the overflow event occurs
<i>TIMER_UPDATECTL_CCUUNDER</i>	the shadow registers update by when the underflow event occurs
<i>TIMER_UPDATECTL_CCUOVERUNDER</i>	the shadow registers update by when the overflow or underflow event occurs
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel control shadow register update when CMTG bit is set */
timer_channel_control_shadow_update_config(TIMER0, TIMER_UPDATECTL_CCU);
```

## timer\_channel\_output\_struct\_para\_init

The description of timer\_channel\_output\_struct\_para\_init is shown as below:

**Table 3-1171. 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 3-1172. 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* ocpa);
<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=0~4, 7, 14~16, 22, 23, 40~44)</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, 22, 23, 40~44))
<i>TIMER_CH_1</i>	TIMER channel 1 (TIMERx(x=0~4, 7, 14, 22, 23, 40~44))

<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<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_ocinitpara.outputstate = TIMER_CCX_ENABLE;

timer_ocinitpara.outputnstate = TIMER_CCXN_ENABLE;

timer_ocinitpara.ocpolarity = TIMER_OC_POLARITY_HIGH;

timer_ocinitpara.ocnpolarity = TIMER_OCN_POLARITY_HIGH;

timer_ocinitpara.ocidlestate = TIMER_OC_IDLE_STATE_HIGH;

timer_ocinitpara.ocnidlestate = TIMER_OCN_IDLE_STATE_LOW;

timer_channel_output_config(TIMER0, TIMER_CH_0, &timer_ocinitpara);

```

### timer\_channel\_output\_mode\_config

The description of timer\_channel\_output\_mode\_config is shown as below:

**Table 3-1173. 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=0~4,7,14~16,22,23,40~44)</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=0~4,7,14~16,22,23,40~44))
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)

<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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
<i>TIMER_OC_MODE_ACTIVE</i>	set the channel output
<i>TIMER_OC_MODE_INACTIVE</i>	clear the channel output
<i>TIMER_OC_MODE_TOGGLE</i>	toggle on match
<i>TIMER_OC_MODE_LOW</i>	force low mode
<i>TIMER_OC_MODE_HIGH</i>	force high mode
<i>TIMER_OC_MODE_PWM0</i>	PWM mode 0
<i>TIMER_OC_MODE_PWM1</i>	PWM mode 1
<i>TIMER_OC_MODE_DSPM0</i>	delayable SPM mode 0
<i>TIMER_OC_MODE_DSPM1</i>	delayable SPM mode 1
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel PWM 0 mode */
```

```
timer_channel_output_mode_config(TIMER0, TIMER_CH_0, TIMER_OC_MODE_PWM0);
```

### timer\_channel\_output\_pulse\_value\_config

The description of timer\_channel\_output\_pulse\_value\_config is shown as below:

**Table 3-1174. 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</i> ( <i>x</i> =0~4,7,14~16,22,23,40~44)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0( <i>TIMERx</i> , <i>x</i> =0~4,7,14~16,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1( <i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2( <i>TIMERx</i> , <i>x</i> =0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3( <i>TIMERx</i> , <i>x</i> =0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0( <i>TIMERx</i> , <i>x</i> =0,7,14~16,40~44)
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1( <i>TIMERx</i> , <i>x</i> =0,7)
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2( <i>TIMERx</i> , <i>x</i> =0,7)
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3( <i>TIMERx</i> , <i>x</i> =0,7)
<b>Input parameter{in}</b>	
<b>pulse</b>	channel output pulse value(0~65535)
<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 3-1175. Function timer\_channel\_output\_shadow\_config**

<b>Function name</b>	timer_channel_output_shadow_config
<b>Function prototype</b>	void timer_channel_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t ocshadow);
<b>Function descriptions</b>	configure TIMER channel output shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,14~16,22,23,40~44)	please refer to the following parameters

Input parameter{in}	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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>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
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*configure TIMER0 channel 0 output shadow function */
```

```
timer_channel_output_shadow_config (TIMER0, TIMER_CH_0,  
TIMER_OC_SHADOW_ENABLE);
```

### timer\_channel\_output\_clear\_config

The description of timer\_channel\_output\_clear\_config is shown as below:

**Table 3-1176. 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>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER periphera
<i>TIMERx(x=0~4,7,14~16,</i>	please refer to the following parameters

22,23,40~44)	
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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 3-1177. 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</i> ( <i>x</i> =0~4,7,14~16,22,23,40~44)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0( <i>TIMERx</i> , <i>x</i> =0~4,7,14~16,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1( <i>TIMERx</i> , <i>x</i> =0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2( <i>TIMERx</i> , <i>x</i> =0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3( <i>TIMERx</i> , <i>x</i> =0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0( <i>TIMERx</i> , <i>x</i> =0,7,14~16,40~44)
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1( <i>TIMERx</i> , <i>x</i> =0,7)
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2( <i>TIMERx</i> , <i>x</i> =0,7)
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3( <i>TIMERx</i> , <i>x</i> =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 3-1178. 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 ocpolarity);
<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</i> ( <i>x</i> =0~4, 7, 14~16, 22, 23, 40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0( <i>TIMERx</i> , <i>x</i> =0~4, 7, 14~16, 22, 23, 40~44)
<i>TIMER_CH_1</i>	TIMER channel 1( <i>TIMERx</i> , <i>x</i> =0~4, 7, 14, 22, 23, 40~44)
<i>TIMER_CH_2</i>	TIMER channel 2( <i>TIMERx</i> , <i>x</i> =0~4, 7, 22, 23)
<i>TIMER_CH_3</i>	TIMER channel 3( <i>TIMERx</i> , <i>x</i> =0~4, 7, 22, 23)
<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 3-1179. Function timer\_channel\_output\_state\_config**

<b>Function name</b>	timer_channel_output_state_config
<b>Function prototype</b>	void timer_channel_output_state_config(uint32_t timer_periph, uint16_t channel, uint32_t state);
<b>Function descriptions</b>	configure TIMER channel enable state
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4, 7, 14~16, 22, 23, 40~44)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0( <i>TIMERx</i> , <i>x</i> =0~4, 7, 14~16, 22, 23, 40~44)
<i>TIMER_CH_1</i>	TIMER channel 1( <i>TIMERx</i> , <i>x</i> =0~4, 7, 14, 22, 23, 40~44)

<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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
<i>E</i>	
<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 3-1180. 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=0,7,14~16,40~44)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0(TIMERx, x=0,7,14~16,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0,7,14,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0,7)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0,7)
<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 3-1181. 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 3-1182. Function timer\_input\_capture\_config

Function name	timer_input_capture_config
Function prototype	void timer_input_capture_config(uint32_t timer_periph, uint16_t channel, timer_ic_parameter_struct* icpara);
Function descriptions	configure TIMER input capture parameter
Precondition	-
The called functions	timer_channel_input_capture_prescaler_config
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14~16,22,23,40~44)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0(TIMERx, x=0~4,7,14~16,22,23,40~44)
TIMER_CH_1	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
TIMER_CH_2	TIMER channel 2(TIMERx, x=0~4,7,22,23)
TIMER_CH_3	TIMER channel 3(TIMERx, x=0~4,7,22,23)
TIMER_MCH_0	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
TIMER_MCH_1	TIMER multi mode channel 1(TIMERx, x=0,7)
TIMER_MCH_2	TIMER multi mode channel 2(TIMERx, x=0,7)
TIMER_MCH_3	TIMER multi mode channel 3(TIMERx, x=0,7)
Input parameter{in}	
icpara	TIMER channel input parameter struct, the structure members can refer to <a href="#">Structure timer_ic_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 input capture parameter */
timer_ic_parameter_struct timer_icinitpara;

timer_icinitpara.icpolarity = TIMER_IC_POLARITY_RISING;

timer_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 3-1183. Function timer\_channel\_input\_capture\_prescaler\_config**

<b>Function name</b>	timer_channel_input_capture_prescaler_config
<b>Function prototype</b>	void timer_channel_input_capture_prescaler_config(uint32_t timer_periph, uint16_t channel, uint16_t prescaler);
<b>Function descriptions</b>	configure TIMER channel input capture prescaler 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> =0~4,7,14~16,22,23,40~44)	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=0~4,7,14~16,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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>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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

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 3-1184. 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=0~4,7,14~16,22,23,40~44)</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=0~4,7,14~16,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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 3-1185. 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=0~4,7,14~16,22,23,40~44)</i>	TIMER peripheral selection

Input parameter{in}	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
Input parameter{in}	
<b>icpwm</b>	TIMER channel input PWM parameter struct, the structure members can refer to <a href="#">Structure timer_ic_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TIMER0 input pwm capture parameter */

timer_ic_parameter_struct timer_icinitpara;

timer_icinitpara.icpolarity = TIMER_IC_POLARITY_RISING;

timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTTI;

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 3-1186. 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>	-
Input parameter{in}	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4, 7, 14~16, 22, 23, 40~44)</i>	TIMER peripheral selection
Input parameter{in}	
<b>hallmode</b>	TIMER hall sensor mode state
<i>TIMER_HALLINTERFA CE_ENABLE</i>	TIMER hall sensor mode enable
<i>TIMER_HALLINTERFA CE_DISABLE</i>	TIMER hall sensor mode disable



Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 hall sensor mode */
timer_hall_mode_config (TIMER0, TIMER_HALLINTERFACE_ENABLE);
```

### timer\_multi\_mode\_channel\_output\_parameter\_struct\_init

The description of timer\_multi\_mode\_channel\_output\_parameter\_struct\_init is shown as below:

**Table 3-1187. Function timer\_multi\_mode\_channel\_output\_parameter\_struct\_init**

Function name	timer_multi_mode_channel_output_parameter_struct_init
Function prototype	void timer_multi_mode_channel_output_parameter_struct_init(timer_omc_parameter_struct *omcpara);
Function descriptions	initialize TIMER multi mode channel output parameter struct
Precondition	-
The called functions	-
Input parameter{in}	
omcpara	TIMER multi mode channel output parameter struct, the structure members can refer to <a href="#">Structure timer_omc_parameter_struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER multi mode channel output parameter struct with a default value */
timer_omc_parameter_struct timer_omcinitpara;
timer_multi_mode_channel_output_parameter_struct_init(&timer_omcinitpara);
```

### timer\_multi\_mode\_channel\_output\_config

The description of timer\_multi\_mode\_channel\_output\_config is shown as below:

**Table 3-1188. Function timer\_multi\_mode\_channel\_output\_config**

Function name	timer_multi_mode_channel_output_config
Function prototype	void timer_multi_mode_channel_output_config(uint32_t timer_periph, uint16_t channel, timer_omc_parameter_struct *omcpara);
Function descriptions	configure TIMER multi mode channel output function

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_MCH_0</i>	TIMER multi mode channel 0(TIMERx, x=0,7,14~16,40~44)
<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>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 3-1189. Function timer\_multi\_mode\_channel\_mode\_config**

<b>Function name</b>	timer_multi_mode_channel_mode_config
<b>Function prototype</b>	void timer_multi_mode_channel_mode_config(uint32_t timer_periph, uint32_t channel, uint32_t multi_mode_sel);
<b>Function descriptions</b>	multi mode channel mode select
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral

<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	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>TIMERx</i> , <i>x</i> =0,7,14~16,40~44)
<i>TIMER_MCH_1</i>	TIMER multi mode channel 1( <i>TIMERx</i> , <i>x</i> =0,7)
<i>TIMER_MCH_2</i>	TIMER multi mode channel 2( <i>TIMERx</i> , <i>x</i> =0,7)
<i>TIMER_MCH_3</i>	TIMER multi mode channel 3( <i>TIMERx</i> , <i>x</i> =0,7)
<b>Input parameter{in}</b>	
<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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0 multi mode channel 0 mode */
```

```
timer_multi_mode_channel_mode_config      (TIMER0,          TIMER_MCH_0,
timer_multi_mode_channel_mode_config      (TIMER0,          TIMER_MCH_MODE_INDEPENDENTLY);
```

### timer\_input\_trigger\_source\_select

The description of timer\_input\_trigger\_source\_select is shown as below:

**Table 3-1190. Function timer\_input\_trigger\_source\_select**

<b>Function name</b>	timer_input_trigger_source_select
<b>Function prototype</b>	void timer_input_trigger_source_select(uint32_t timer_periph, uint32_t intrigger);
<b>Function descriptions</b>	select TIMER input trigger source
<b>Precondition</b>	SMC[2:0] = 000
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>intrigger</b>	input trigger source
<i>TIMER_SMCFG_TRGS_EL_ITI0</i>	internal trigger input 0 (ITI0, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44))
<i>TIMER_SMCFG_TRGS</i>	internal trigger input 1 (ITI1, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44))

<i>EL_ITI1</i>	
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI2</i>	internal trigger input 2 (ITI2, $TIMERx(x=0\sim4,7,14,22,23,40\sim44)$ )
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI3</i>	internal trigger input 3 (ITI3, $TIMERx(x=0\sim4,7,14,22,23,40\sim44)$ )
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOF_ED</i>	TI0 edge detector (CIOF_ED, $TIMERx(x=0\sim4,7,14,22,23,40\sim44)$ )
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOFE0</i>	filtered channel channel 0 input (CIOFE0, $TIMERx(x=0\sim4,7,14,22,23,40\sim44)$ )
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI1FE1</i>	filtered channel channel 1 input (CI1FE1, $TIMERx(x=0\sim4,7,14,22,23,40\sim44)$ )
<i>TIMER_SMCFG_TRGS</i> <i>EL_ETIFP</i>	external trigger input filter output(ETIFP, $TIMERx(x=0\sim4,7,22,23)$ )
<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( $TIMERx(x=0,7,14,40\sim44)$ )
<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( $TIMERx(x=0,7)$ )
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI4</i>	internal trigger 4 for General-L0 timer( $TIMERx(x=1,2,22,23)$ )
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI5</i>	internal trigger 5 for General-L0 timer( $TIMERx(x=1,22,23)$ )
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI6</i>	internal trigger 6 for General-L0 timer
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI7</i>	internal trigger 7 for General-L0 timer
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI8</i>	internal trigger 8 for General-L0 timer
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI9</i>	internal trigger 9 for General-L0 timer( $TIMERx(x=22,23)$ )
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI10</i>	internal trigger 10 for General-L0 timer( $TIMERx(x=22,23)$ )
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI11</i>	internal trigger 11 for General-L0 timer( $TIMERx(x=22,23)$ )
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI12</i>	internal trigger input 12( $TIMERx(x=0\sim4,7,23)$ )

<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI13</i>	internal trigger input 13(TIMERx(x=0~4,7,22))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI14</i>	internal trigger input 14(TIMERx(x=0~4,7,14,22,23,40~44))
<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\_output0\_trigger\_source\_select

The description of timer\_master\_output0\_trigger\_source\_select is shown as below:

**Table 3-1191. Function timer\_master\_output0\_trigger\_source\_select**

<b>Function name</b>	timer_master_output0_trigger_source_select
<b>Function prototype</b>	void timer_master_output0_trigger_source_select(uint32_t timer_periph, uint32_t outrigger);
<b>Function descriptions</b>	select TIMER master mode output 0 trigger source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14,22,23,50,51)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>outrigger</b>	trigger output source
<i>TIMER_TRI_OUT0_SR</i> <i>C_RESET</i>	the UPG bit as trigger output 0 (TIMERx(x=0~7,14,22,23,50,51))
<i>TIMER_TRI_OUT0_SR</i> <i>C_ENABLE</i>	the counter enable signal as trigger output 0(TIMERx(x=0~7,14,22,23,50,51))
<i>TIMER_TRI_OUT0_SR</i> <i>C_UPDATE</i>	update event as trigger output 0(TIMERx(x=0~7,14,22,23,50,51))
<i>TIMER_TRI_OUT0_SR</i> <i>C_CH0</i>	a capture or a compare match occurred in channel 0 as trigger output 0 (TIMERx(x=0~4,7,14,22,23,40~44))
<i>TIMER_TRI_OUT0_SR</i> <i>C_O0CPRE</i>	O0CPRE as trigger output 0(TIMERx(x=0~4,7,14,22,23,40~44))
<i>TIMER_TRI_OUT0_SR</i> <i>C_O1CPRE</i>	O1CPRE as trigger output 0(TIMERx(x=0~4,7,14,22,23,40~44))
<i>TIMER_TRI_OUT0_SR</i> <i>C_O2CPRE</i>	O2CPRE as trigger output 0(TIMERx(x=0~4,7,22,23))

<i>TIMER_TRI_OUT0_SR</i> <i>C_O3CPRE</i>	O3CPRE as trigger output 0(TIMERx(x=0~4,7,22,23))
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0 master mode output 0 trigger source */
```

```
timer_master_output0_trigger_source_select (TIMER0, TIMER_TRI_OUT0_SRC_RESET);
```

### timer\_master\_output1\_trigger\_source\_select

The description of timer\_master\_output1\_trigger\_source\_select is shown as below:

**Table 3-1192. Function timer\_master\_output1\_trigger\_source\_select**

<b>Function name</b>	timer_master_output1_trigger_source_select
<b>Function prototype</b>	void timer_master_output1_trigger_source_select(uint32_t timer_periph, uint32_t outrigger);
<b>Function descriptions</b>	select TIMER master mode output 1 trigger source
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>outrigger</b>	trigger output source
<i>TIMER_TRI_OUT1_SR</i> <i>C_RESET</i>	the UPG bit as trigger output 1
<i>TIMER_TRI_OUT1_SR</i> <i>C_ENABLE</i>	the counter enable signal as trigger output 1
<i>TIMER_TRI_OUT1_SR</i> <i>C_UPDATE</i>	update event as trigger output 1
<i>TIMER_TRI_OUT1_SR</i> <i>C_CH0</i>	a capture or a compare match occurred in channel 0 as trigger output 1
<i>TIMER_TRI_OUT1_SR</i> <i>C_O0CPRE</i>	O0CPRE as trigger output 1
<i>TIMER_TRI_OUT1_SR</i> <i>C_O1CPRE</i>	O1CPRE as trigger output 1
<i>TIMER_TRI_OUT1_SR</i> <i>C_O2CPRE</i>	O2CPRE as trigger output 1
<i>TIMER_TRI_OUT1_SR</i> <i>C_O3CPRE</i>	O3CPRE as trigger output 1
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```
/* select TIMER0 master mode output 1 trigger source */
```

```
timer_master_output1_trigger_source_select (TIMER0, TIMER_TRI_OUT1_SRC_RESET);
```

### timer\_slave\_mode\_select

The description of timer\_slave\_mode\_select is shown as below:

**Table 3-1193. 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
TIMERx(x=0~4,7,14,22,23,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>slavemode</b>	slave mode
TIMER_SLAVE_MODE_DISABLE	slave mode disable(TIMERx(x=0~4,7,14,22,23,40~44))
TIMER_ENCODER_MODE0	encoder mode 0(TIMERx(x=0~4,7,22,23))
TIMER_ENCODER_MODE1	encoder mode 1(TIMERx(x=0~4,7,22,23))
TIMER_ENCODER_MODE2	encoder mode 2(TIMERx(x=0~4,7,22,23))
TIMER_SLAVE_MODE_RESTART	restart mode(TIMERx(x=0~4,7,14,22,23,40~44))
TIMER_SLAVE_MODE_PAUSE	pause mode(TIMERx(x=0~4,7,14,22,23,40~44))
TIMER_SLAVE_MODE_EVENT	event mode(TIMERx(x=0~4,7,14,22,23,40~44))
TIMER_SLAVE_MODE_EXTERNAL0	external clock mode 0(TIMERx(x=0~4,7,14,22,23,40~44))
TIMER_SLAVE_MODE_RESTART_EVENT	restart + event mode(TIMERx(x=0~4,7,14,22,23,40~44))
TIMER_NONQUAD_MODE0	non-quadrature decoder mode 0(TIMERx(x=0~4,7,22,23))

<i>TIMER_NONQUAD_MODE1</i>	non-quadrature decoder mode 1(TIMERx(x=0~4,7,22,23))
<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 3-1194. 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=0~4,7,14,22,23,40~44)</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 3-1195. Function timer\_external\_trigger\_config

Function name	timer_external_trigger_config
Function prototype	void timer_external_trigger_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
Function descriptions	configure TIMER external trigger input
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,22,23)	TIMER peripheral selection
Input parameter{in}	
extprescaler	external trigger prescaler
TIMER_EXT_TRI_PSC_OFF	no divided
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
Input parameter{in}	
expolarity	external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active
TIMER_ETP_RISING	active high or rising edge active
Input parameter{in}	
extfilter	external trigger filter control(0~15)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure TIMER0 external trigger input */
```

```
timer_external_trigger_config(TIMER0, TIMER_EXT_TRI_PSC_DIV2,
TIMER_ETP_FALLING, 10);
```

### timer\_quadrature\_decoder\_mode\_config

The description of timer\_quadrature\_decoder\_mode\_config is shown as below:

Table 3-1196. Function timer\_quadrature\_decoder\_mode\_config

Function name	timer_quadrature_decoder_mode_config
Function prototype	void timer_quadrature_decoder_mode_config(uint32_t timer_periph, uint32_t decompode, uint16_t ic0polarity, uint16_t ic1polarity);

<b>Function descriptions</b>	configure TIMER quadrature decoder mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,22,23)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>decomode</b>	quadrature decoder mode
<i>TIMER_ENCODER_MODE0</i>	counter counts on CI0FE0 edge depending on CI1FE1 level
<i>TIMER_ENCODER_MODE1</i>	counter counts on CI1FE1 edge depending on CI0FE0 level
<i>TIMER_ENCODER_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\_non\_quadrature\_decoder\_mode\_config

The description of timer\_non\_quadrature\_decoder\_mode\_config is shown as below:

**Table 3-1197. Function timer\_non\_quadrature\_decoder\_mode\_config**

<b>Function name</b>	timer_non_quadrature_decoder_mode_config
<b>Function prototype</b>	void timer_non_quadrature_decoder_mode_config(uint32_t timer_periph, uint32_t decomode, uint16_t ic1polarity);
<b>Function descriptions</b>	configure TIMER non-quadrature decoder mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,22,23)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>decomode</b>	quadrature decoder mode
<i>TIMER_NONQUAD_MODE0</i>	non-quadrature decoder mode 0
<i>TIMER_NONQUAD_MODE1</i>	non-quadrature decoder mode 1
<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
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 non-quadrature decoder mode */
```

```
timer_non_quadrature_decoder_mode_config (TIMER0, TIMER_NONQUAD_MODE0,  
TIMER_IC_POLARITY_RISING);
```

### timer\_internal\_clock\_config

The description of timer\_internal\_clock\_config is shown as below:

**Table 3-1198. 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</i> ( <i>x</i> =0~4,7,14,22,23,40~44)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 internal clock mode */
```

```
timer_internal_clock_config (TIMER0);
```

### timer\_internal\_trigger\_as\_external\_clock\_config

The description of timer\_internal\_trigger\_as\_external\_clock\_config is shown as below:

**Table 3-1199. Function timer\_internal\_trigger\_as\_external\_clock\_config**

<b>Function name</b>	timer_internal_trigger_as_external_clock_config
<b>Function prototype</b>	void timer_internal_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t intrigger);
<b>Function descriptions</b>	configure TIMER the internal trigger as external clock input
<b>Precondition</b>	-
<b>The called functions</b>	timer_input_trigger_source_select
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>intrigger</b>	trigger selection
<i>TIMER_SMCFG_TRGS_EL_ITI0</i>	internal trigger input 0 (ITI0, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI1</i>	internal trigger input 1 (ITI1, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI2</i>	internal trigger input 2 (ITI2, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44))
<i>TIMER_SMCFG_TRGS_EL_ITI3</i>	internal trigger input 3 (ITI3, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23,40~44))
<i>TIMER_L0_SMCFG_TR_GSEL_ITI4</i>	internal trigger 4 for General-L0 timer( <i>TIMERx</i> ( <i>x</i> =1,2,22,23))
<i>TIMER_L0_SMCFG_TR_GSEL_ITI5</i>	internal trigger 5 for General-L0 timer( <i>TIMERx</i> ( <i>x</i> =1,22,23))
<i>TIMER_L0_SMCFG_TR_GSEL_ITI7</i>	internal trigger 7 for General-L0 timer( <i>TIMERx</i> ( <i>x</i> =4))
<i>TIMER_L0_SMCFG_TR_GSEL_ITI9</i>	internal trigger 9 for General-L0 timer( <i>TIMERx</i> ( <i>x</i> =22,23))

<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI10</i>	internal trigger 10 for General-L0 timer(TIMERx(x=22,23))
<i>TIMER_L0_SMCFG_TR</i> <i>GSEL_ITI11</i>	internal trigger 11 for General-L0 timer(TIMERx(x=22,23))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI12</i>	internal trigger input 12(TIMERx(x=0~4,7,23))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI13</i>	internal trigger input 13(TIMERx(x=0~4,7,22))
<i>TIMER_SMCFG_TRGS</i> <i>EL_ITI14</i>	internal trigger input 14(TIMERx(x=0~4,7,14,22,23,40~44))
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 the internal trigger ITI0 as external clock input */
```

```
timer_internal_trigger_as_external_clock_config(TIMER0, TIMER_SMCFG_TRGSEL_ITI0);
```

### timer\_external\_trigger\_as\_external\_clock\_config

The description of timer\_external\_trigger\_as\_external\_clock\_config is shown as below:

**Table 3-1200. Function timer\_external\_trigger\_as\_external\_clock\_config**

<b>Function name</b>	timer_external_trigger_as_external_clock_config
<b>Function prototype</b>	void timer_external_trigger_as_external_clock_config(uint32_t timer_periph, uint32_t extrigger, uint16_t expolarity, uint32_t extfilter);
<b>Function descriptions</b>	configure TIMER the external trigger as external clock input
<b>Precondition</b>	-
<b>The called functions</b>	timer_input_trigger_source_select
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,40~44)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>extrigger</b>	external trigger selection
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOF_ED</i>	TI0 edge detector (CIOF_ED, TIMERx(x=0~4,7,14,22,23,40~44))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CIOFE0</i>	filtered channel channel 0 input (CIOFE0, TIMERx(x=0~4,7,14,22,23,40~44))
<i>TIMER_SMCFG_TRGS</i> <i>EL_CI1FE1</i>	filtered channel channel 1 input (CI1FE1, TIMERx(x=0~4,7,14,22,23,40~44))
<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=0,7,14,40~44))
<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>Input parameter{in}</b>	
<b>expolarity</b>	external trigger polarity
<i>TIMER_IC_POLARITY_</i> <i>RISING</i>	active high or rising edge active
<i>TIMER_IC_POLARITY_</i> <i>FALLING</i>	active low or falling edge active
<i>TIMER_IC_POLARITY_</i> <i>BOTH_EDGE</i>	active both edge
<b>Input parameter{in}</b>	
<b>extfilter</b>	external trigger filter control(0~15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 the external trigger CI0FE0 as external clock input */
```

```
timer_external_trigger_as_external_clock_config(TIMER0,  
TIMER_SMCFG_TRGSEL_CI0FE0, TIMER_IC_POLARITY_RISING, 0);
```

### timer\_external\_clock\_mode0\_config

The description of timer\_external\_clock\_mode0\_config is shown as below:

**Table 3-1201. Function timer\_external\_clock\_mode0\_config**

<b>Function name</b>	timer_external_clock_mode0_config
<b>Function prototype</b>	void timer_external_clock_mode0_config(uint32_t timer_periph, uint32_t extprescaler, uint32_t expolarity, uint32_t extfilter);
<b>Function descriptions</b>	configure TIMER the external clock mode0
<b>Precondition</b>	-
<b>The called functions</b>	timer_external_trigger_config
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,</i>	TIMER peripheral selection

23,40~44)	
<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 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 3-1202. 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=0~4,7,22,23)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>extprescaler</b>	ETI external trigger prescaler
<i>TIMER_EXT_TRI_PSC_</i>	no divided

OFF	
TIMER_EXT_TRI_PSC_DIV2	divided by 2
TIMER_EXT_TRI_PSC_DIV4	divided by 4
TIMER_EXT_TRI_PSC_DIV8	divided by 8
<b>Input parameter{in}</b>	
<b>expolarity</b>	ETI external trigger polarity
TIMER_ETP_FALLING	active low or falling edge active
TIMER_ETP_RISING	active high or rising edge active
<b>Input parameter{in}</b>	
<b>extfilter</b>	ETI external trigger filter control(0~15)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 the external clock mode1 */
```

```
timer_external_clock_mode1_config(TIMER0, TIMER_EXT_TRI_PSC_DIV2,  
TIMER_ETP_FALLING, 0);
```

### timer\_external\_clock\_mode1\_disable

The description of timer\_external\_clock\_mode1\_disable is shown as below:

**Table 3-1203. Function timer\_external\_clock\_mode1\_disable**

<b>Function name</b>	timer_external_clock_mode1_disable
<b>Function prototype</b>	void timer_external_clock_mode1_disable(uint32_t timer_periph);
<b>Function descriptions</b>	disable TIMER the external clock mode1
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
TIMERx(x=0~4,7,22,23)	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 3-1204. 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=0~4, 14~16, 22, 23, 40~44)</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>	
-	-
<b>Return value</b>	
-	-

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 3-1205. Function timer\_output\_value\_selection\_config**

<b>Function name</b>	timer_output_value_selection_config
<b>Function prototype</b>	void timer_output_value_selection_config(uint32_t timer_periph, uint16_t outsel);
<b>Function descriptions</b>	configure TIMER output value selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral

<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>outsel</b>	output value selection
<i>TIMER_OUTSEL_DISABLE</i>	no effect
<i>TIMER_OUTSEL_ENABLE</i>	if POEN and IOS is 0, the output disabled
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER output value selection */
```

```
timer_output_value_selection_config(TIMER0, TIMER_OUTSEL_ENABLE);
```

### timer\_commutation\_control\_shadow\_register\_config

The description of timer\_commutation\_control\_shadow\_register\_config is shown as below:

**Table 3-1206. Function timer\_commutation\_control\_shadow\_register\_config**

<b>Function name</b>	timer_commutation_control_shadow_register_config
<b>Function prototype</b>	void timer_commutation_control_shadow_register_config(uint32_t timer_periph, uint16_t ccssel);
<b>Function descriptions</b>	configure commutation control shadow register update selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>ccsel</b>	commutation control shadow register selection
<i>TIMER_CCUSEL_ENABLE</i>	the shadow registers update when the counter generates an overflow/underflow event
<i>TIMER_CCUSEL_DISABLE</i>	the shadow registers update when the counter generates an overflow/underflow event and the repetition counter value is zero
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure commutation control shadow register update selection */
```

```
timer_commutation_control_shadow_register_config (TIMER0, TIMER_CCUSEL_ENABLE);
```

### timer\_output\_match\_pulse\_select

The description of timer\_output\_match\_pulse\_select is shown as below:

**Table 3-1207. Function timer\_output\_match\_pulse\_select**

<b>Function name</b>	timer_output_match_pulse_select
<b>Function prototype</b>	void timer_output_match_pulse_select(uint32_t timer_periph, uint32_t channel, uint16_t pulsesel);
<b>Function descriptions</b>	configure TIMER output match pulse selection
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,40~44)</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=0~4,7,14,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<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\_mode\_config

The description of timer\_channel\_composite\_pwm\_mode\_config is shown as below:

**Table 3-1208. Function timer\_channel\_composite\_pwm\_mode\_config**

<b>Function name</b>	timer_channel_composite_pwm_mode_config
<b>Function prototype</b>	void timer_channel_composite_pwm_mode_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure the TIMER composite PWM mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,14,22, 23,40~44)	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=0~4,7,14,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER composite PWM mode */
```

```
timer_channel_composite_pwm_mode_config (TIMER0, TIMER_CH_0, ENABLE);
```

## timer\_channel\_composite\_pwm\_mode\_output\_pulse\_value\_config

The description of timer\_channel\_composite\_pwm\_mode\_output\_pulse\_value\_config is shown as below:

**Table 3-1209. Function timer\_channel\_composite\_pwm\_mode\_output\_pulse\_value\_config**

<b>Function name</b>	timer_channel_composite_pwm_mode_output_pulse_value_config
<b>Function prototype</b>	void timer_channel_composite_pwm_mode_output_pulse_value_config(uint32_t

	t timer_periph, uint32_t channel, uint32_t pulse, uint32_t add_pulse);
<b>Function descriptions</b>	configure the TIMER composite PWM mode output pulse value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,40~44)</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=0~4,7,14,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<b>Input parameter{in}</b>	
<b>pulse</b>	channel compare value(0~0xFFFFFFFF)
<b>Input parameter{in}</b>	
<b>add_pulse</b>	channel additional compare value(0~0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel 0 output pulse value */
```

```
timer_channel_output_pulse_value_config(TIMER0, TIMER_CH_0, 399, 3999);
```

### timer\_channel\_additional\_compare\_value\_config

The description of timer\_channel\_additional\_compare\_value\_config is shown as below:

**Table 3-1210. Function timer\_channel\_additional\_compare\_value\_config**

<b>Function name</b>	timer_channel_additional_compare_value_config
<b>Function prototype</b>	void timer_channel_additional_compare_value_config(uint32_t timer_periph, uint16_t channel, uint32_t value);
<b>Function descriptions</b>	configure TIMER channel additional compare value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,40~44)</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=0~4,7,14,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<b>Input parameter{in}</b>	
<b>value</b>	channel additional compare value(0~0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel 0 additional compare value */
```

```
timer_channel_additional_compare_value_config (TIMER0, TIMER_CH_0, 399);
```

### timer\_channel\_additional\_output\_shadow\_config

The description of timer\_channel\_additional\_output\_shadow\_config is shown as below:

**Table 3-1211. Function timer\_channel\_additional\_output\_shadow\_config**

<b>Function name</b>	timer_channel_additional_output_shadow_config
<b>Function prototype</b>	void timer_channel_additional_output_shadow_config(uint32_t timer_periph, uint16_t channel, uint16_t aocshadow);
<b>Function descriptions</b>	configure TIMER channel additional output shadow function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,14,22,23,40~44)</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=0~4,7,14,22,23,40~44)
<i>TIMER_CH_1</i>	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
<i>TIMER_CH_2</i>	TIMER channel 2(TIMERx, x=0~4,7,22,23)
<i>TIMER_CH_3</i>	TIMER channel 3(TIMERx, x=0~4,7,22,23)
<b>Input parameter{in}</b>	
<b>aocshadow</b>	channel additional output compare shadow
<i>TIMER_ADD_SHADOW_ENABLE</i>	channel additional output compare shadow enable
<i>TIMER_ADD_SHADOW_DISABLE</i>	channel additional output compare shadow disable
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/*configure TIMER0 channel 0 additional output shadow function */
```

```
timer_channel_additional_output_shadow_config      (TIMER0,          TIMER_CH_0,
TIMER_OC_SHADOW_ENABLE);
```

### timer\_channel\_additional\_compare\_value\_read

The description of timer\_channel\_additional\_compare\_value\_read is shown as below:

**Table 3-1212. Function timer\_channel\_additional\_compare\_value\_read**

Function name	timer_channel_additional_compare_value_read
Function prototype	uint32_t timer_channel_additional_compare_value_read(uint32_t timer_periph, uint16_t channel);
Function descriptions	read TIMER channel additional compare value
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~4,7,14,22,23,40~44)	TIMER peripheral selection
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0(TIMERx, x=0~4,7,14,22,23,40~44)
TIMER_CH_1	TIMER channel 1(TIMERx, x=0~4,7,14,22,23,40~44)
TIMER_CH_2	TIMER channel 2(TIMERx, x=0~4,7,22,23)
TIMER_CH_3	TIMER channel 3(TIMERx, x=0~4,7,22,23)
Output parameter{out}	
-	-
Return value	
uint32_t	channel additional compare value, 0~0xFFFFFFFF

Example:

```
/* get TIMER autoreload register value */
```

```
uint32_t i = 0;
```

```
i =timer_channel_additional_compare_value_read(TIMER0, TIMER_CH_0);
```

### timer\_break\_external\_source\_config

The description of timer\_break\_external\_source\_config is shown as below:

Table 3-1213. Function timer\_break\_external\_source\_config

Function name	timer_break_external_source_config
Function prototype	void timer_break_external_source_config(uint32_t timer_periph, uint16_t break_num, uint32_t break_src, ControlStatus newvalue);
Function descriptions	configure the TIMER break source
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7,14~16,40~44)	please refer to the following parameters
Input parameter{in}	
break_num	TIMER BREAKx
TIMER_BREAK0	BREAK0 input signals, TIMERx(x=0,7,14~16,40~44)
TIMER_BREAK1	BREAK1 input signals, TIMERx(x=0,7)
Input parameter{in}	
break_src	break source
TIMER_BRKIN0	BRKIN0 alternate function input enable, TIMERx(x=0,7,14~16,40~44)
TIMER_BRKIN1	BRKIN1 alternate function input enable, TIMERx(x=0,7)
TIMER_BRKIN2	BRKIN2 alternate function input enable, TIMERx(x=0,7)
TIMER_BRKCOMP0	CMP0 input enable, TIMERx(x=0,7,14~16,40~44)
TIMER_BRKCOMP1	CMP1 input enable, TIMERx(x=0,7,14~16,40~44)
TIMER_BRKHPDF	HPDF input enable, TIMERx(x=0,7,14~16,40~44)
Input parameter{in}	
newvalue	control value
ENABLE	enable function
DISABLE	disable function
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure the TIMER break source */
```

```
timer_break_external_source_config(TIMER0,    TIMER_BREAK0,    TIMER_BRKIN0,
ENABLE);
```

### timer\_break\_external\_polarity\_config

The description of timer\_break\_external\_polarity\_config is shown as below:

Table 3-1214. Function timer\_break\_external\_polarity\_config

Function name	timer_break_external_polarity_config
Function prototype	void timer_break_external_polarity_config(uint32_t timer_periph, uint16_t



	break_num, uint32_t break_src, uint16_t bkinpolarity);
<b>Function descriptions</b>	configure TIMER break polarity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7,14~16,40~44)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b><i>break_num</i></b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	BREAK0 input signals, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_BREAK1</i>	BREAK1 input signals, TIMERx(x=0,7)
<b>Input parameter{in}</b>	
<b>break_src</b>	break source
<i>TIMER_BRKIN0</i>	BRKIN0 alternate function input enable, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_BRKIN1</i>	BRKIN1 alternate function input enable, TIMERx(x=0,7)
<i>TIMER_BRKIN2</i>	BRKIN2 alternate function input enable, TIMERx(x=0,7)
<i>TIMER_BRKCOMP0</i>	CMP0 input enable, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_BRKCOMP1</i>	CMP1 input enable, TIMERx(x=0,7,14~16,40~44)
<b>Input parameter{in}</b>	
<b>bkinpolarity</b>	break polarity
<i>TIMER_BRKIN_POLARITY_LOW</i>	active low
<i>TIMER_BRKIN_POLARITY_HIGH</i>	active high
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TIMER break polarity */
```

```
timer_break_external_polarity_config(TIMER0,    TIMER_BREAK0,    TIMER_BRKIN0,
TIMER BRKIN POLARITY HIGH);
```

## timer\_break\_lock\_config

The description of timer break lock config is shown as below:

### Table 3-1215. Function timer break lock config

<b>Function name</b>	timer_break_lock_config
<b>Function prototype</b>	void timer_break_lock_config(uint32_t timer_periph, uint16_t break_num, ControlStatus newvalue);
<b>Function descriptions</b>	configure TIMER break lock function

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>break_num</b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	enable BREAK0 lock function, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)
<i>TIMER_BREAK1</i>	enable BREAK1 lock function, <i>TIMERx</i> ( <i>x</i> =0,7)
<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 TIMER break lock function */
```

```
timer_break_lock_config(TIMER0, TIMER_BREAK0, ENABLE);
```

### timer\_break\_lock\_release\_config

The description of timer\_break\_lock\_release\_config is shown as below:

**Table 3-1216. Function timer\_break\_lock\_release\_config**

<b>Function name</b>	timer_break_lock_release_config
<b>Function prototype</b>	void timer_break_lock_release_config(uint32_t timer_periph, uint16_t break_num, ControlStatus newvalue);
<b>Function descriptions</b>	release the TIMER break lock function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>break_num</b>	TIMER BREAKx
<i>TIMER_BREAK0</i>	release the BREAK0 lock function, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)
<i>TIMER_BREAK1</i>	release the BREAK1 lock function, <i>TIMERx</i> ( <i>x</i> =0,7)
<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:

```
/* release the TIMER break lock function */
```

```
timer_break_lock_release_config (TIMER0, TIMER_BREAK0, ENABLE);
```

### timer\_channel\_break\_control\_config

The description of timer\_channel\_break\_control\_config is shown as below:

**Table 3-1217. Function timer\_channel\_break\_control\_config**

<b>Function name</b>	timer_channel_break_control_config
<b>Function prototype</b>	void timer_channel_break_control_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure the TIMER channel break function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2
<i>TIMER_CH_3</i>	TIMER channel 3
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel 0 break function */
```

```
timer_channel_break_control_config (TIMER0, TIMER_CH_0, ENABLE);
```

## timer\_channel\_dead\_time\_config

The description of timer\_channel\_dead\_time\_config is shown as below:

**Table 3-1218. Function timer\_channel\_dead\_time\_config**

<b>Function name</b>	timer_channel_dead_time_config
<b>Function prototype</b>	void timer_channel_dead_time_config(uint32_t timer_periph, uint32_t channel, ControlStatus newvalue);
<b>Function descriptions</b>	configure the TIMER channel free dead time function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0,7)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>channel</b>	TIMER channel
<i>TIMER_CH_0</i>	TIMER channel 0
<i>TIMER_CH_1</i>	TIMER channel 1
<i>TIMER_CH_2</i>	TIMER channel 2
<i>TIMER_CH_3</i>	TIMER channel 3
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TIMER0 channel 0 free dead time function */
timer_channel_dead_time_config (TIMER0, TIMER_CH_0, ENABLE);
```

## timer\_free\_complementary\_struct\_para\_init

The description of timer\_free\_complementary\_struct\_para\_init is shown as below:

**Table 3-1219. Function timer\_free\_complementary\_struct\_para\_init**

<b>Function name</b>	timer_free_complementary_struct_para_init
<b>Function prototype</b>	void timer_free_complementary_struct_para_init(timer_free_complementary_parameter_struct *freecompara);
<b>Function descriptions</b>	initialize TIMER channel free complementary parameter struct with a default value

Precondition	-
The called functions	-
Input parameter{in}	
freecompara	TIMER channel free complementary parameter struct, the structure members can refer to <a href="#">Structure timer free complementary parameter struct</a> .
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* initialize TIMER channel free complementary parameter struct with a default value */
timer_free_complementary_parameter_struct timer_freecompara;
timer_free_complementary_struct_para_init (&timer_freecompara);
```

### timer\_channel\_free\_complementary\_config

The description of timer\_channel\_free\_complementary\_config is shown as below:

**Table 3-1220. Function timer\_channel\_free\_complementary\_config**

Function name	timer_channel_free_complementary_config
Function prototype	void timer_channel_free_complementary_config(uint32_t timer_periph, uint16_t channel, timer_free_complementary_parameter_struct *fcpa);
Function descriptions	configure channel free complementary protection
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0,7)	please refer to the following parameters
Input parameter{in}	
channel	TIMER channel
TIMER_CH_0	TIMER channel 0
TIMER_CH_1	TIMER channel 1
TIMER_CH_2	TIMER channel 2
TIMER_CH_3	TIMER channel 3
Input parameter{in}	
freecompara	TIMER channel free complementary parameter struct, the structure members can refer to <a href="#">Structure timer free complementary parameter struct</a> .
Output parameter{out}	
-	-
Return value	

-	-
---	---

Example:

```

/* initialize TIMER break parameter struct with a default value */

timer_free_complementary_parameter_struct timer_freecompara;

timer_freecompara.freecomstate = TIMER_FCCHP_STATE_ENABLE;

timer_freecompara.runoffstate   = TIMER_ROS_STATE_ENABLE;

timer_freecompara.ideloffstate  = TIMER_IOS_STATE_ENABLE;

timer_freecompara.deadtime      = 255;

timer_channel_free_complementary_config(&timer_freecompara);

```

### timer\_watchdog\_value\_config

The description of timer\_watchdog\_value\_config is shown as below:

**Table 3-1221. Function timer\_watchdog\_value\_config**

<b>Function name</b>	timer_watchdog_value_config
<b>Function prototype</b>	void timer_watchdog_value_config(uint32_t timer_periph, uint32_t value);
<b>Function descriptions</b>	configure TIMER autoreload register value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4, 7, 22, 23)</i>	TIMER peripheral selection
<b>Input parameter{in}</b>	
<b>value</b>	watchdog counter period value, 0~0xFFFFFFFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure quadrature decoder signal disconnection detection watchdog value */

timer_watchdog_value_config(TIMER0, 3000);

```

### timer\_watchdog\_value\_read

The description of timer\_watchdog\_value\_read is shown as below:

**Table 3-1222. Function timer\_watchdog\_value\_read**

<b>Function name</b>	timer_watchdog_value_read
<b>Function prototype</b>	uint32_t timer_watchdog_value_read(uint32_t timer_periph);

<b>Function descriptions</b>	read quadrature decoder signal disconnection detection watchdog value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,22,23)	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint32_t</b>	watchdog counter period register value, 0~0xFFFFFFFF

Example:

```

/* read quadrature decoder signal disconnection detection watchdog value */

uint32_t i = 0;

i =timer_watchdog_value_read(TIMER0);

```

### timer\_decoder\_disconnection\_detection\_config

The description of timer\_decoder\_disconnection\_detection\_config is shown as below:

**Table 3-1223. Function timer\_decoder\_disconnection\_detection\_config**

<b>Function name</b>	timer_decoder_disconnection_detection_config
<b>Function prototype</b>	void timer_decoder_disconnection_detection_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure quadrature decoder signal disconnection detection function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx</i> ( <i>x</i> =0~4,7,22,23)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure quadrature decoder signal disconnection detection function */

timer_decoder_disconnection_detection_config(TIMER0, ENABLE);

```

## timer\_decoder\_jump\_detection\_config

The description of timer\_decoder\_jump\_detection\_config is shown as below:

**Table 3-1224. Function timer\_decoder\_jump\_detection\_config**

<b>Function name</b>	timer_decoder_jump_detection_config
<b>Function prototype</b>	void timer_decoder_jump_detection_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure quadrature decoder signal jump detection function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~4,7,22,23)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value
<i>ENABLE</i>	enable function
<i>DISABLE</i>	disable function
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure quadrature decoder signal jump detection function */
timer_decoder_jump_detection_config(TIMER0, ENABLE);
```

## timer\_upif\_backup\_config

The description of timer\_upif\_backup\_config is shown as below:

**Table 3-1225. Function timer\_upif\_backup\_config**

<b>Function name</b>	timer_upif_backup_config
<b>Function prototype</b>	void timer_upif_backup_config(uint32_t timer_periph, ControlStatus newvalue);
<b>Function descriptions</b>	configure the UPIF bit backup function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,40~44,50,51)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>newvalue</b>	control value



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

Example:

```
/* configure the UPIF bit backup function */
```

```
timer_upif_backup_config(TIMER0, ENABLE);
```

### timer\_upifbu\_bit\_get

The description of timer\_upifbu\_bit\_get is shown as below:

**Table 3-1226. Function timer\_upifbu\_bit\_get**

<b>Function name</b>	timer_upifbu_bit_get
<b>Function prototype</b>	UPIFBUSStatus timer_upifbu_bit_get(uint32_t timer_periph);
<b>Function descriptions</b>	get the UPIFBU bit in the TIMEx_CNT register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMEx(x=0~7, 14~16, 22, 23, 40~44, 50, 51)</i>	TIMER peripheral selection
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>UPIFBUSStatus</b>	UPIFBU bit state
<i>VALID_SET</i>	the UPIFBU is valid and value is 1
<i>VALID_RESET</i>	the UPIFBU is valid and value is 0
<i>INVALID</i>	the UPIFBU is invalid

Example:

```
/* get the UPIFBU bit in the TIMEx_CNT register */
```

```
UPIFBUSStatus upstatus = INVALID;
```

```
upstatus = timer_upifbu_bit_get (TIMER0);
```

### timer\_flag\_get

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

**Table 3-1227. 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=0~7,14~16,22,23,40~44,50,51)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>flag</b>	the TIMER flags
<i>TIMER_FLAG_UP</i>	update flag, <i>TIMERx(x=0~7,14~16,22,23,40~44,50,51)</i>
<i>TIMER_FLAG_CH0</i>	channel 0 capture or compare flag, <i>TIMERx(x=0~4,7,14~16,22,23,40~44)</i>
<i>TIMER_FLAG_CH1</i>	channel 1 capture or compare flag, <i>TIMERx(x=0~4,7,14,22,23,40~44)</i>
<i>TIMER_FLAG_CH2</i>	channel 2 capture or compare flag, <i>TIMERx(x=0~4,7,22,23)</i>
<i>TIMER_FLAG_CH3</i>	channel 3 capture or compare flag, <i>TIMERx(x=0~4,7,22,23)</i>
<i>TIMER_FLAG_CMT</i>	channel commutation flag, <i>TIMERx(x=0,7,14~16,40~44)</i>
<i>TIMER_FLAG_TRG</i>	trigger flag, <i>TIMERx(x=0~4,7,14,22,23,40~44)</i>
<i>TIMER_FLAG_BRK0</i>	BREAK0 flag, <i>TIMERx(x=0,7,14~16,40~44)</i>
<i>TIMER_FLAG_BRK1</i>	BREAK1 flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_CH0O</i>	channel 0 overcapture flag, <i>TIMERx(x=0~4,7,14~16,22,23,40~44)</i>
<i>TIMER_FLAG_CH1O</i>	channel 1 overcapture flag, <i>TIMERx(x=0~4,7,14,22,23,40~44)</i>
<i>TIMER_FLAG_CH2O</i>	channel 2 overcapture flag, <i>TIMERx(x=0~4,7,22,23)</i>
<i>TIMER_FLAG_CH3O</i>	channel 3 overcapture flag, <i>TIMERx(x=0~4,7,22,23)</i>
<i>TIMER_FLAG_SYSB</i>	system source break flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_DECJ</i>	quadrature decoder signal jump flag, <i>TIMERx(x=0~4,7,22,23)</i>
<i>TIMER_FLAG_DECDIS</i>	quadrature decoder signal disconnection flag, <i>TIMERx(x=0~4,7,22,23)</i>
<i>TIMER_FLAG_MCH0</i>	multi mode channel 0 capture or compare flag, <i>TIMERx(x=0,7,14~16,40~44)</i>
<i>TIMER_FLAG_MCH1</i>	multi mode channel 1 capture or compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH2</i>	multi mode channel 2 capture or compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH3</i>	multi mode channel 3 capture or compare flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH0O</i>	multi mode channel 0 overcapture flag, <i>TIMERx(x=0,7,14~16,40~44)</i>
<i>TIMER_FLAG_MCH1O</i>	multi mode channel 1 overcapture flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH2O</i>	multi mode channel 2 overcapture flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_MCH3O</i>	multi mode channel 3 overcapture flag, <i>TIMERx(x=0,7)</i>
<i>TIMER_FLAG_CH0CO MADD</i>	channel 0 additional compare flag, <i>TIMERx(x=0~4,7,14,22,23)</i>
<i>TIMER_FLAG_CH1CO MADD</i>	channel 1 additional compare flag, <i>TIMERx(x=0~4,7,14,22,23)</i>
<i>TIMER_FLAG_CH2CO MADD</i>	channel 2 additional compare flag, <i>TIMERx(x=0~4,7,22,23)</i>
<i>TIMER_FLAG_CH3CO MADD</i>	channel 3 additional compare flag, <i>TIMERx(x=0~4,7,22,23)</i>

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

Example:

```
/* get TIMER0 update flags */
```

```
FlagStatus Flag_status = RESET;
```

```
Flag_status = timer_flag_get(TIMER0, TIMER_FLAG_UP);
```

### timer\_flag\_clear

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

**Table 3-1228. Function timer\_flag\_clear**

Function name	timer_flag_clear
Function prototype	void timer_flag_clear(uint32_t timer_periph, uint32_t flag);
Function descriptions	clear TIMER flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7,14~16,22,23,40~44,50,51)	please refer to the following parameters
Input parameter{in}	
flag	the timer interrupt flags
TIMER_FLAG_UP	update flag, TIMERx(x=0~7,14~16,22,23,40~44,50,51)
TIMER_FLAG_CH0	channel 0 capture or compare flag, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_FLAG_CH1	channel 1 capture or compare flag, TIMERx(x=0~4,7,14,22,23,40~44)
TIMER_FLAG_CH2	channel 2 capture or compare flag, TIMERx(x=0~4,7,22,23)
TIMER_FLAG_CH3	channel 3 capture or compare flag, TIMERx(x=0~4,7,22,23)
TIMER_FLAG_CMT	channel commutation flag, TIMERx(x=0,7,14~16,40~44)
TIMER_FLAG_TRG	trigger flag, TIMERx(x=0~4,7,14,22,23,40~44)
TIMER_FLAG_BRK0	BREAK0 flag, TIMERx(x=0,7,14~16,40~44)
TIMER_FLAG_BRK1	BREAK1 flag, TIMERx(x=0,7)
TIMER_FLAG_CH0O	channel 0 overcapture flag, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_FLAG_CH1O	channel 1 overcapture flag, TIMERx(x=0~4,7,14,22,23,40~44)
TIMER_FLAG_CH2O	channel 2 overcapture flag, TIMERx(x=0~4,7,22,23)
TIMER_FLAG_CH3O	channel 3 overcapture flag, TIMERx(x=0~4,7,22,23)
TIMER_FLAG_SYSB	system source break flag, TIMERx(x=0,7)
TIMER_FLAG_DECJ	quadrature decoder signal jump flag, TIMERx(x=0~4,7,22,23)
TIMER_FLAG_DECDIS	quadrature decoder signal disconnection flag, TIMERx(x=0~4,7,22,23)
TIMER_FLAG_MCH0	multi mode channel 0 capture or compare flag,

	TIMERx(x=0,7,14~16,40~44)
<i>TIMER_FLAG_MCH1</i>	multi mode channel 1 capture or compare flag, TIMERx(x=0,7)
<i>TIMER_FLAG_MCH2</i>	multi mode channel 2 capture or compare flag, TIMERx(x=0,7)
<i>TIMER_FLAG_MCH3</i>	multi mode channel 3 capture or compare flag, TIMERx(x=0,7)
<i>TIMER_FLAG_MCH0O</i>	multi mode channel 0 overcapture flag, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_FLAG_MCH1O</i>	multi mode channel 1 overcapture flag, TIMERx(x=0,7)
<i>TIMER_FLAG_MCH2O</i>	multi mode channel 2 overcapture flag, TIMERx(x=0,7)
<i>TIMER_FLAG_MCH3O</i>	multi mode channel 3 overcapture flag, TIMERx(x=0,7)
<i>TIMER_FLAG_CH0CO MADD</i>	channel 0 additional compare flag, TIMERx(x=0~4,7,14,22,23)
<i>TIMER_FLAG_CH1CO MADD</i>	channel 1 additional compare flag, TIMERx(x=0~4,7,14,22,23)
<i>TIMER_FLAG_CH2CO MADD</i>	channel 2 additional compare flag, TIMERx(x=0~4,7,22,23)
<i>TIMER_FLAG_CH3CO MADD</i>	channel 3 additional compare flag, TIMERx(x=0~4,7,22,23)
<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 3-1229. Function timer\_interrupt\_enable**

<b>Function name</b>	timer_interrupt_enable
<b>Function prototype</b>	void timer_interrupt_enable(uint32_t timer_periph, uint32_t interrupt);
<b>Function descriptions</b>	enable the TIMER interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>timer_periph</b>	TIMER peripheral
<i>TIMERx(x=0~7,14~16,22,23,40~44,50,51)</i>	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>interrupt</b>	timer interrupt enable source
<i>TIMER_INT_UP</i>	update interrupt enable, TIMERx(x=0~7,14~16,22,23,40~44,50,51)
<i>TIMER_INT_CH0</i>	channel 0 capture or compare interrupt enable, TIMERx(x=0~4,7,14~16,22,23,40~44)

<i>TIMER_INT_CH1</i>	channel 1 capture or compare interrupt enable, TIMERx(x=0~4,7,14,22,23,40~44)
<i>TIMER_INT_CH2</i>	channel 2 capture or compare interrupt enable, TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_CH3</i>	channel 3 capture or compare interrupt enable , TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_CMT</i>	commutation interrupt enable, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_INT_TRG</i>	trigger interrupt enable, TIMERx(x=0~4,7,14,22,23,40~44)
<i>TIMER_INT_BRK</i>	break interrupt enable, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_INT_DECJ</i>	quadrature decoder signal jump interrupt, TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_DECDIS</i>	quadrature decoder signal disconnection interrupt, TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_MCH0</i>	multi mode channel 0 capture or compare interrupt, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_INT_MCH1</i>	multi mode channel 1 capture or compare interrupt, TIMERx(x=0,7)
<i>TIMER_INT_MCH2</i>	multi mode channel 2 capture or compare interrupt, TIMERx(x=0,7)
<i>TIMER_INT_MCH3</i>	multi mode channel 3 capture or compare interrupt, TIMERx(x=0,7)
<i>TIMER_INT_CH0COMA DD</i>	channel 0 additional compare interrupt, TIMERx(x=0~4,7,14,22,23,40~44)
<i>TIMER_INT_CH0COMA DD</i>	channel 1 additional compare interrupt, TIMERx(x=0~4,7,14,22,23,40~44)
<i>TIMER_INT_CH0COMA DD</i>	channel 2 additional compare interrupt, TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_CH0COMA DD</i>	channel 3 additional compare interrupt, TIMERx(x=0~4,7,22,23)
<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 3-1230. 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=0~7,14~16,2</i>	please refer to the following parameters

2,23,40~44,50,51)	
Input parameter{in}	
<b>interrupt</b>	timer interrupt disable source
<i>TIMER_INT_UP</i>	update interrupt enable, TIMERx(x=0~7,14~16,22,23,40~44,50,51)
<i>TIMER_INT_CH0</i>	channel 0 capture or compare interrupt enable, TIMERx(x=0~4,7,14~16,22,23,40~44)
<i>TIMER_INT_CH1</i>	channel 1 capture or compare interrupt enable, TIMERx(x=0~4,7,14,22,23,40~44)
<i>TIMER_INT_CH2</i>	channel 2 capture or compare interrupt enable, TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_CH3</i>	channel 3 capture or compare interrupt enable , TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_CMT</i>	commutation interrupt enable, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_INT_TRG</i>	trigger interrupt enable, TIMERx(x=0~4,7,14,22,23,40~44)
<i>TIMER_INT_BRK</i>	break interrupt enable, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_INT_DECJ</i>	quadrature decoder signal jump interrupt, TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_DECDIS</i>	quadrature decoder signal disconnection interrupt, TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_MCH0</i>	multi mode channel 0 capture or compare interrupt, TIMERx(x=0,7,14~16,40~44)
<i>TIMER_INT_MCH1</i>	multi mode channel 1 capture or compare interrupt, TIMERx(x=0,7)
<i>TIMER_INT_MCH2</i>	multi mode channel 2 capture or compare interrupt, TIMERx(x=0,7)
<i>TIMER_INT_MCH3</i>	multi mode channel 3 capture or compare interrupt, TIMERx(x=0,7)
<i>TIMER_INT_CH0COMA DD</i>	channel 0 additional compare interrupt, TIMERx(x=0~4,7,14,22,23,40~44)
<i>TIMER_INT_CH0COMA DD</i>	channel 1 additional compare interrupt, TIMERx(x=0~4,7,14,22,23,40~44)
<i>TIMER_INT_CH0COMA DD</i>	channel 2 additional compare interrupt, TIMERx(x=0~4,7,22,23)
<i>TIMER_INT_CH0COMA DD</i>	channel 3 additional compare interrupt, TIMERx(x=0~4,7,22,23)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TIMER0 update interrupt */
```

```
timer_interrupt_disable(TIMER0, TIMER_INT_UP);
```

### timer\_interrupt\_flag\_get

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

**Table 3-1231. 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
TIMERx(x=0~7,14~16,22,23,40~44,50,51)	please refer to the following parameters
<b>Input parameter{in}</b>	
<b>int_flag</b>	the timer interrupt bits
TIMER_INT_FLAG_UP	update interrupt flag, TIMERx(x=0~7,14~16,22,23,40~44,50,51)
TIMER_INT_FLAG_CH0	channel 0 capture or compare interrupt flag, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_INT_FLAG_CH1	channel 1 capture or compare interrupt flag, TIMERx(x=0~4,7,14,22,23,40~44)
TIMER_INT_FLAG_CH2	channel 2 capture or compare interrupt flag, TIMERx(x=0~4,7,22,23)
TIMER_INT_FLAG_CH3	channel 3 capture or compare interrupt flag, TIMERx(x=0~4,7,22,23)
TIMER_INT_FLAG_CMT	channel commutation interrupt flag, TIMERx(x=0,7,14~16,40~44)
TIMER_INT_FLAG_TRG	trigger interrupt flag, TIMERx(x=0~4,7,14,22,23,40~44)
TIMER_INT_FLAG_BRK0	BREAK0 interrupt flag, TIMERx(x=0,7,14~16,40~44)
TIMER_INT_FLAG_BRK1	BREAK1 interrupt flag, TIMERx(x=0,7)
TIMER_INT_FLAG_SYSB	system source break interrupt flag, TIMERx(x=0,7)
TIMER_INT_FLAG_DECJ	quadrature decoder signal jump interrupt flag, TIMERx(x=0~4,7,22,23)
TIMER_INT_FLAG_DECDIS	quadrature decoder signal disconnection interrupt flag, TIMERx(x=0~4,7,22,23)
TIMER_INT_FLAG_MCH0	multi mode channel 0 capture or compare interrupt flag, TIMERx(x=0,7,14~16,40~44)
TIMER_INT_FLAG_MCH1	multi mode channel 1 capture or compare interrupt flag, TIMERx(x=0,7)
TIMER_INT_FLAG_MCH2	multi mode channel 2 capture or compare interrupt flag, TIMERx(x=0,7)
TIMER_INT_FLAG_MCH3	multi mode channel 3 capture or compare interrupt flag, TIMERx(x=0,7)
TIMER_INT_FLAG_CH0COMADD	channel 0 additional compare interrupt flag, TIMERx(x=0~4,7,14,22,23)
TIMER_INT_FLAG_CH1COMADD	channel 1 additional compare interrupt flag, TIMERx(x=0~4,7,14,22,23)
TIMER_INT_FLAG_CH2	channel 2 additional compare interrupt flag, TIMERx(x=0~4,7,22,23)

COMADD	
TIMER_INT_FLAG_CH3 COMADD	channel 3 additional compare interrupt flag, TIMERx(x=0~4,7,22,23)
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get TIMER0 update interrupt flag */
```

```
FlagStatus Flag_interrupt = RESET;
```

```
Flag_interrupt = timer_interrupt_flag_get(TIMER0, TIMER_INT_FLAG_UP);
```

### timer\_interrupt\_flag\_clear

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

**Table 3-1232. Function timer\_interrupt\_flag\_clear**

Function name	timer_interrupt_flag_clear
Function prototype	void timer_interrupt_flag_clear(uint32_t timer_periph, uint32_t int_flag);
Function descriptions	clear TIMER interrupt flags
Precondition	-
The called functions	-
Input parameter{in}	
timer_periph	TIMER peripheral
TIMERx(x=0~7,14~16,22,23,40~44,50,51)	please refer to the following parameters
Input parameter{in}	
int_flag	the timer interrupt bits
TIMER_INT_FLAG_UP	update interrupt flag, TIMERx(x=0~7,14~16,22,23,40~44,50,51)
TIMER_INT_FLAG_CH0	channel 0 capture or compare interrupt flag, TIMERx(x=0~4,7,14~16,22,23,40~44)
TIMER_INT_FLAG_CH1	channel 1 capture or compare interrupt flag, TIMERx(x=0~4,7,14,22,23,40~44)
TIMER_INT_FLAG_CH2	channel 2 capture or compare interrupt flag, TIMERx(x=0~4,7,22,23)
TIMER_INT_FLAG_CH3	channel 3 capture or compare interrupt flag, TIMERx(x=0~4,7,22,23)
TIMER_INT_FLAG_CMT	channel commutation interrupt flag, TIMERx(x=0,7,14~16,40~44)
TIMER_INT_FLAG_TRG	trigger interrupt flag, TIMERx(x=0~4,7,14,22,23,40~44)
TIMER_INT_FLAG_BRK0	BREAK0 interrupt flag, TIMERx(x=0,7,14~16,40~44)
TIMER_INT_FLAG_BRK1	BREAK1 interrupt flag, TIMERx(x=0,7)



<i>TIMER_INT_FLAG_SYS</i> <i>B</i>	system source break interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_INT_FLAG_DEC</i> <i>J</i>	quadrature decoder signal jump interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,22,23)
<i>TIMER_INT_FLAG_DEC</i> <i>DIS</i>	quadrature decoder signal disconnection interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,22,23)
<i>TIMER_INT_FLAG_MC</i> <i>H0</i>	multi mode channel 0 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7,14~16,40~44)
<i>TIMER_INT_FLAG_MC</i> <i>H1</i>	multi mode channel 1 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_INT_FLAG_MC</i> <i>H2</i>	multi mode channel 2 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_INT_FLAG_MC</i> <i>H3</i>	multi mode channel 3 capture or compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0,7)
<i>TIMER_INT_FLAG_CH0</i> <i>COMADD</i>	channel 0 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23)
<i>TIMER_INT_FLAG_CH1</i> <i>COMADD</i>	channel 1 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,14,22,23)
<i>TIMER_INT_FLAG_CH2</i> <i>COMADD</i>	channel 2 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,22,23)
<i>TIMER_INT_FLAG_CH3</i> <i>COMADD</i>	channel 3 additional compare interrupt flag, <i>TIMERx</i> ( <i>x</i> =0~4,7,22,23)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear TIMER0 update interrupt flag */
```

```
timer_interrupt_flag_clear(TIMER0, TIMER_INT_FLAG_UP);
```

### 3.33. 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 [3.33.1](#), the TMU firmware functions are introduced in chapter [3.33.2](#).

#### 3.33.1. Descriptions of Peripheral registers

TMU registers are listed in the table shown as below:

**Table 3-1233. TMU Registers**

Registers	Descriptions
TMU_CS	TMU control and status register
TMU_IDATA	TMU input data register
TMU_ODATA	TMU output data register

### 3.33.2. Descriptions of Peripheral functions

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

**Table 3-1234. 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_read_interrupt_enable	enable TMU read interrupt
tmu_read_interrupt_disable	disable TMU read interrupt
tmu_dma_read_enable	enable TMU DMA read request
tmu_dma_read_disable	disable TMU DMA read request
tmu_dma_write_enable	enable TMU DMA write request
tmu_dma_write_disable	disable TMU DMA write request
tmu_one_q31_write	write one data in q1.31 format
tmu_two_q31_write	write two data in q1.31 format
tmu_two_q15_write	write two data in q1.15 format
tmu_one_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

### Structure tmu\_parameter\_struct

**Table 3-1235. Structure tmu\_parameter\_struct**

Member name	Function description
mode	mode of TMU operation(TMU_MODE_COS, TMU_MODE_SIN, TMU_MODE_ATAN2, TMU_MODE_MODULUS, TMU_MODE_ATAN, TMU_MODE_COSH, TMU_MODE_SINH, TMU_MODE_ATANH, TMU_MODE_LN, TMU_MODE_SQRT)
iterations_number	number of iterations selection(TMU_ITERATION_STEPS_x(x=4,8,12,...24))
scale	scaling factor(TMU_SCALING_FACTOR_x(x=1,2,4,8,16,32,64,128))
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)

Member name	Function description
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 3-1236. Function tmu\_deinit**

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

Example:

```
/* reset TMU */
tmu_deinit();
```

## tmu\_struct\_para\_init

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

**Table 3-1237. Function tmu\_struct\_para\_init**

Function name	tmu_struct_para_init
Function prototype	void tmu_struct_para_init(timer_parameter_struct* initpara);
Function descriptions	initialize the parameters of TMU struct with the default values
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 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 3-1238. Function tmu\_init**

<b>Function name</b>	tmu_init
<b>Function prototype</b>	void tmu_init(tmu_parameter_struct* init_struct);
<b>Function descriptions</b>	initialize TMU
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>init_struct</b>	pointer to TMU init parameter struct, the structure members can refer to <a href="#">Structure tmu_parameter_struct</a> .
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* initialize TMU */

tmu_parameter_struct tmu_init_struct;

tmu_init_struct.mode = TMU_MODE_COS;

tmu_init_struct.iterations_number = TMU_ITERATION_STEPS_24;

tmu_init_struct.scale = TMU_SCALING_FACTOR_1;

tmu_init_struct.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\_read\_interrupt\_enable

The description of tmu\_read\_interrupt\_enable is shown as below:

**Table 3-1239. Function tmu\_read\_interrupt\_enable**

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

Example:

```
/* enable TMU read interrupt */
tmu_read_interrupt_enable();
```

## tmu\_read\_interrupt\_disable

The description of tmu\_read\_interrupt\_disable is shown as below:

**Table 3-1240. Function tmu\_read\_interrupt\_disable**

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

Example:

```
/* disable TMU read interrupt */
tmu_read_interrupt_disable();
```

## tmu\_dma\_read\_enable

The description of tmu\_dma\_read\_enable is shown as below:

**Table 3-1241. Function tmu\_dma\_read\_enable**

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

Example:

```
/* enable TMU DMA read request */
```

```
tmu_dma_read_enable();
```

### tmu\_dma\_read\_disable

The description of tmu\_dma\_read\_disable is shown as below:

**Table 3-1242. Function tmu\_dma\_read\_disable**

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

Example:

```
/* disable TMU DMA read request */
```

```
tmu_dma_read_disable();
```

### tmu\_dma\_write\_enable

The description of tmu\_dma\_write\_enable is shown as below:

**Table 3-1243. Function tmu\_dma\_write\_enable**

<b>Function name</b>	tmu_dma_write_enable
----------------------	----------------------

<b>Function prototype</b>	void tmu_dma_write_enable(void);
<b>Function descriptions</b>	enable TMU write interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable TMU DMA write request */
tmu_dma_write_enable();
```

### tmu\_dma\_write\_disable

The description of tmu\_dma\_write\_disable is shown as below:

**Table 3-1244. 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 3-1245. 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 3-1246. Function tmu\_two\_q31\_write**

<b>Function name</b>	tmu_two_q31_write
<b>Function prototype</b>	void tmu_two_q31_write(uint32_t data1, uint32_t data2);
<b>Function descriptions</b>	write two data in q1.31 format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>data1</b>	the first input data in q1.31 format (0x00000000~0xFFFFFFFF)
<b>Input parameter{in}</b>	
<b>data2</b>	the second input data in q1.31 format (0x00000000~0xFFFFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write two data in q1.31 format */
```

```
int32_t in1 = -2000;
```

```
int32_t in2 = 3000;
```

```
tmu_two_q31_write((uint32_t)in1, (uint32_t)in2);
```

### tmu\_two\_q15\_write

The description of tmu\_two\_q15\_write is shown as below:



Table 3-1247. Function `tmu_two_q15_write`

Function name	<code>tmu_two_q15_write</code>
Function prototype	<code>void tmu_two_q15_write(uint16_t data1, uint16_t data2);</code>
Function descriptions	write two data in q1.15 format
Precondition	-
The called functions	-
Input parameter{in}	
<b>data1</b>	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\_q31\_read**

The description of `tmu_one_q31_read` is shown as below:

Table 3-1248. Function `tmu_one_q31_read`

Function name	<code>tmu_one_q31_read</code>
Function prototype	<code>void tmu_one_q31_read(uint32_t* p);</code>
Function descriptions	read one data in q1.31 format
Precondition	-
The called functions	-
Input parameter{in}	
<b>p</b>	a pointer to output data(q1.31 format)
Output parameter{out}	
-	-
Return value	
-	-

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 3-1249. Function tmu\_two\_q31\_read**

<b>Function name</b>	tmu_two_q31_read
<b>Function prototype</b>	void tmu_two_q31_read(uint32_t* p1, uint32_t* p2);
<b>Function descriptions</b>	read two data in q1.31 format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>p1</b>	a pointer to the first output data(q1.31 format)
<b>Input parameter{in}</b>	
<b>p2</b>	a pointer to the second output data(q1.31 format)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* read two data in q1.31 format */
uint32_t out1 = 0;
uint32_t out2 = 0;
tmu_two_q31_read(&out1, &out2);
```

## tmu\_two\_q15\_read

The description of tmu\_two\_q15\_read is shown as below:

**Table 3-1250. Function tmu\_two\_q15\_read**

<b>Function name</b>	tmu_two_q15_read
<b>Function prototype</b>	void tmu_two_q15_read(uint16_t* p1, uint16_t* p2);
<b>Function descriptions</b>	read two data in q1.15 format
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>p1</b>	a pointer to the first output data(q1.15 format)
<b>Input parameter{in}</b>	
<b>p2</b>	a pointer to the second output data(q1.15 format)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* read two data in q1.15 format */

uint16_t out1 = 0;

uint16_t out2 = 0;

tmu_two_q15_read(&out1, &out2);
```

## 3.34. TRIGSEL

TRIGSEL is the trigger selection controller in the MCU. It allows software to select the trigger input signal for various peripherals. The TRIGSEL registers are listed in chapter [3.34.1](#), the TRIGSEL firmware functions are introduced in chapter [3.34.2](#).

### 3.34.1. Descriptions of Peripheral registers

TRIGSEL registers are listed in the table shown as below:

**Table 3-1251. TRIGSEL Registers**

Registers	Descriptions
TRIGSEL_EXTOUT0	TRIGSEL trigger selection for EXTOUT0 register
TRIGSEL_EXTOUT1	TRIGSEL trigger selection for EXTOUT1 register
TRIGSEL_EXTOUT2	TRIGSEL trigger selection for EXTOUT2 register
TRIGSEL_EXTOUT3	TRIGSEL trigger selection for EXTOUT3 register
TRIGSEL_ADC0	TRIGSEL trigger selection for ADC0 register
TRIGSEL_ADC1	TRIGSEL trigger selection for ADC1 register
TRIGSEL_ADC2	TRIGSEL trigger selection for ADC2 register
TRIGSEL_DACOUT0	TRIGSEL trigger selection for DAC_OUT0 register
TRIGSEL_DACOUT1	TRIGSEL trigger selection for DAC_OUT1 register
TRIGSEL_TIMER0BRKIN	TRIGSEL trigger selection for TIMER0_BRKIN register
TRIGSEL_TIMER7BRKIN	TRIGSEL trigger selection for TIMER7_BRKIN register
TRIGSEL_TIMER14BRKIN	TRIGSEL trigger selection for TIMER14_BRKIN register
TRIGSEL_TIMER15BRKIN	TRIGSEL trigger selection for TIMER15_BRKIN register
TRIGSEL_TIMER16BRKIN	TRIGSEL trigger selection for TIMER16_BRKIN register
TRIGSEL_TIMER40BRKIN	TRIGSEL trigger selection for TIMER40_BRKIN register
TRIGSEL_TIMER41BRKIN	TRIGSEL trigger selection for TIMER41_BRKIN register
TRIGSEL_TIMER42BRKIN	TRIGSEL trigger selection for TIMER42_BRKIN register
TRIGSEL_TIMER43BRKIN	TRIGSEL trigger selection for TIMER43_BRKIN register
TRIGSEL_TIMER44BRKIN	TRIGSEL trigger selection for TIMER44_BRKIN register
TRIGSEL_CAN0	TRIGSEL trigger selection for CAN0 register
TRIGSEL_CAN1	TRIGSEL trigger selection for CAN1 register
TRIGSEL_CAN2	TRIGSEL trigger selection for CAN2 register
TRIGSEL_LPPTS	TRIGSEL trigger selection for LPPTS register

Registers	Descriptions
TRIGSEL_TIMER0ETI	TRIGSEL trigger selection for TIMER0_ETI register
TRIGSEL_TIMER1ETI	TRIGSEL trigger selection for TIMER1_ETI register
TRIGSEL_TIMER2ETI	TRIGSEL trigger selection for TIMER2_ETI register
TRIGSEL_TIMER3ETI	TRIGSEL trigger selection for TIMER3_ETI register
TRIGSEL_TIMER4ETI	TRIGSEL trigger selection for TIMER4_ETI register
TRIGSEL_TIMER7ETI	TRIGSEL trigger selection for TIMER7_ETI register
TRIGSEL_TIMER22ETI	TRIGSEL trigger selection for TIMER22_ETI register
TRIGSEL_TIMER23ETI	TRIGSEL trigger selection for TIMER23_ETI register
TRIGSEL_EDOUT	TRIGSEL trigger selection for EDOUT register
TRIGSEL_HPDPF	TRIGSEL trigger selection for HPDPF register
TRIGSEL_TIMER0ITI14	TRIGSEL trigger selection for TIMER0_ITI14 register
TRIGSEL_TIMER1ITI14	TRIGSEL trigger selection for TIMER1_ITI14 register
TRIGSEL_TIMER2ITI14	TRIGSEL trigger selection for TIMER2_ITI14 register
TRIGSEL_TIMER3ITI14	TRIGSEL trigger selection for TIMER3_ITI14 register
TRIGSEL_TIMER4ITI14	TRIGSEL trigger selection for TIMER4_ITI14 register
TRIGSEL_TIMER7ITI14	TRIGSEL trigger selection for TIMER7_ITI14 register
TRIGSEL_TIMER14ITI14	TRIGSEL trigger selection for TIMER14_ITI14 register
TRIGSEL_TIMER22ITI14	TRIGSEL trigger selection for TIMER22_ITI14 register
TRIGSEL_TIMER23ITI14	TRIGSEL trigger selection for TIMER23_ITI14 register
TRIGSEL_TIMER40ITI14	TRIGSEL trigger selection for TIMER40_ITI14 register
TRIGSEL_TIMER41ITI14	TRIGSEL trigger selection for TIMER41_ITI14 register
TRIGSEL_TIMER42ITI14	TRIGSEL trigger selection for TIMER42_ITI14 register
TRIGSEL_TIMER43ITI14	TRIGSEL trigger selection for TIMER43_ITI14 register
TRIGSEL_TIMER44ITI14	TRIGSEL trigger selection for TIMER44_ITI14 register

### 3.34.2. Descriptions of Peripheral functions

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

**Table 3-1252. TRIGSEL firmware function**

Function name	Function description
trigsel_deinit	Deinitialize TRIGSEL
trigsel_init	Set the trigger input signal for target peripheral
trigsel_trigger_source_get	Get the trigger input signal for target peripheral
trigsel_register_lock_set	Lock the trigger register
trigsel_register_lock_get	Get the trigger register lock status

#### Enum trigsel\_source\_enum

**Table 3-1253. Enum trigsel\_source\_enum**

Member name	Function description
TRIGSEL_INPUT_0	Trigger input source 0
TRIGSEL_INPUT_1	Trigger input source 1

Member name	Function description
TRIGSEL_INPUT_TRIGSEL_IN0	Trigger input source TRIGSEL_IN0 pin
TRIGSEL_INPUT_TRIGSEL_IN1	Trigger input source TRIGSEL_IN1 pin
TRIGSEL_INPUT_TRIGSEL_IN2	Trigger input source TRIGSEL_IN2 pin
TRIGSEL_INPUT_TRIGSEL_IN3	Trigger input source TRIGSEL_IN3 pin
TRIGSEL_INPUT_TRIGSEL_IN4	Trigger input source TRIGSEL_IN4 pin
TRIGSEL_INPUT_TRIGSEL_IN5	Trigger input source TRIGSEL_IN5 pin
TRIGSEL_INPUT_TRIGSEL_IN6	Trigger input source TRIGSEL_IN6 pin
TRIGSEL_INPUT_TRIGSEL_IN7	Trigger input source TRIGSEL_IN7 pin
TRIGSEL_INPUT_TRIGSEL_IN8	Trigger input source TRIGSEL_IN8 pin
TRIGSEL_INPUT_TRIGSEL_IN9	Trigger input source TRIGSEL_IN9 pin
TRIGSEL_INPUT_TRIGSEL_IN10	Trigger input source TRIGSEL_IN10 pin
TRIGSEL_INPUT_TRIGSEL_IN11	Trigger input source TRIGSEL_IN11 pin
TRIGSEL_INPUT_TRIGSEL_IN12	Trigger input source TRIGSEL_IN12 pin
TRIGSEL_INPUT_TRIGSEL_IN13	Trigger input source TRIGSEL_IN13 pin
TRIGSEL_INPUT_LXTAL_TRG	Trigger input source LXTAL_TRG
TRIGSEL_INPUT_TIMER0_TRGO0	Trigger input source TIMER0 TRGO0
TRIGSEL_INPUT_TIMER0_TRGO1	Trigger input source TIMER0 TRGO1
TRIGSEL_INPUT_TIMER0_CH0	Trigger input source TIMER0 CH0
TRIGSEL_INPUT_TIMER0_CH1	Trigger input source TIMER0 CH1
TRIGSEL_INPUT_TIMER0_CH2	Trigger input source TIMER0 CH2
TRIGSEL_INPUT_TIMER0_CH3	Trigger input source TIMER0 CH3
TRIGSEL_INPUT_TIMER0_MCH0	Trigger input source TIMER0 MCH0
TRIGSEL_INPUT_TIMER0_MCH1	Trigger input source TIMER0 MCH1
TRIGSEL_INPUT_TIMER0_MCH2	Trigger input source TIMER0 MCH2
TRIGSEL_INPUT_TIMER0_MCH3	Trigger input source TIMER0 MCH3
TRIGSEL_INPUT_TIMER0_BRKIN0	Trigger input source TIMER0 BRKIN0
TRIGSEL_INPUT_TIMER0_BRKIN1	Trigger input source TIMER0 BRKIN1
TRIGSEL_INPUT_TIMER0_BRKIN2	Trigger input source TIMER0 BRKIN2
TRIGSEL_INPUT_TIMER0_ETI	Trigger input source TIMER0 ETI
TRIGSEL_INPUT_TIMER1_TRGO0	Trigger input source TIMER1 TRGO0
TRIGSEL_INPUT_TIMER1_CH0	Trigger input source TIMER1 CH0
TRIGSEL_INPUT_TIMER1_CH1	Trigger input source TIMER1 CH1
TRIGSEL_INPUT_TIMER1_CH2	Trigger input source TIMER1 CH2
TRIGSEL_INPUT_TIMER1_CH3	Trigger input source TIMER1 CH3
TRIGSEL_INPUT_TIMER1_ETI	Trigger input source TIMER1 ETI
TRIGSEL_INPUT_TIMER2_TRGO0	Trigger input source TIMER2 TRGO0
TRIGSEL_INPUT_TIMER2_CH0	Trigger input source TIMER2 CH0
TRIGSEL_INPUT_TIMER2_CH1	Trigger input source TIMER2 CH1
TRIGSEL_INPUT_TIMER2_CH2	Trigger input source TIMER2 CH2
TRIGSEL_INPUT_TIMER2_CH3	Trigger input source TIMER2 CH3
TRIGSEL_INPUT_TIMER2_ETI	Trigger input source TIMER2 ETI
TRIGSEL_INPUT_TIMER3_TRGO0	Trigger input source TIMER3 TRGO0

Member name	Function description
TRIGSEL_INPUT_TIMER3_CH0	Trigger input source TIMER3 CH0
TRIGSEL_INPUT_TIMER3_CH1	Trigger input source TIMER3 CH1
TRIGSEL_INPUT_TIMER3_CH2	Trigger input source TIMER3 CH2
TRIGSEL_INPUT_TIMER3_CH3	Trigger input source TIMER3 CH3
TRIGSEL_INPUT_TIMER3_ETI	Trigger input source TIMER3 ETI
TRIGSEL_INPUT_TIMER4_TRGO0	Trigger input source TIMER4 TRGO0
TRIGSEL_INPUT_TIMER4_CH0	Trigger input source TIMER4 CH0
TRIGSEL_INPUT_TIMER4_CH1	Trigger input source TIMER4 CH1
TRIGSEL_INPUT_TIMER4_CH2	Trigger input source TIMER4 CH2
TRIGSEL_INPUT_TIMER4_CH3	Trigger input source TIMER4 CH3
TRIGSEL_INPUT_TIMER4_ETI	Trigger input source TIMER4 ETI
TRIGSEL_INPUT_TIMER5_TRGO0	Trigger input source TIMER5 TRGO0
TRIGSEL_INPUT_TIMER6_TRGO0	Trigger input source TIMER6 TRGO0
TRIGSEL_INPUT_TIMER7_TRGO0	Trigger input source TIMER7 TRGO0
TRIGSEL_INPUT_TIMER7_TRGO1	Trigger input source TIMER7 TRGO1
TRIGSEL_INPUT_TIMER7_CH0	Trigger input source TIMER7 CH0
TRIGSEL_INPUT_TIMER7_CH1	Trigger input source TIMER7 CH1
TRIGSEL_INPUT_TIMER7_CH2	Trigger input source TIMER7 CH2
TRIGSEL_INPUT_TIMER7_CH3	Trigger input source TIMER7 CH3
TRIGSEL_INPUT_TIMER7_MCH0	Trigger input source TIMER7 MCH0
TRIGSEL_INPUT_TIMER7_MCH1	Trigger input source TIMER7 MCH1
TRIGSEL_INPUT_TIMER7_MCH2	Trigger input source TIMER7 MCH2
TRIGSEL_INPUT_TIMER7_MCH3	Trigger input source TIMER7 MCH3
TRIGSEL_INPUT_TIMER7_BRKIN0	Trigger input source TIMER7 BRKIN0
TRIGSEL_INPUT_TIMER7_BRKIN1	Trigger input source TIMER7 BRKIN1
TRIGSEL_INPUT_TIMER7_BRKIN2	Trigger input source TIMER7 BRKIN2
TRIGSEL_INPUT_TIMER7_ETI	Trigger input source TIMER7 ETI
TRIGSEL_INPUT_TIMER14_TRGO0	Trigger input source TIMER14 TRGO0
TRIGSEL_INPUT_TIMER14_CH0	Trigger input source TIMER14 CH0
TRIGSEL_INPUT_TIMER14_CH1	Trigger input source TIMER14 CH1
TRIGSEL_INPUT_TIMER14_MCH0	Trigger input source TIMER14 MCH0
TRIGSEL_INPUT_TIMER14_BRKIN	Trigger input source TIMER14 BRKIN
TRIGSEL_INPUT_TIMER15_CH0	Trigger input source TIMER15 CH0
TRIGSEL_INPUT_TIMER15_MCH0	Trigger input source TIMER15 MCH0
TRIGSEL_INPUT_TIMER15_BRKIN	Trigger input source TIMER15 BRKIN
TRIGSEL_INPUT_TIMER16_CH0	Trigger input source TIMER16 CH0
TRIGSEL_INPUT_TIMER16_MCH0	Trigger input source TIMER16 MCH0
TRIGSEL_INPUT_TIMER16_BRKIN	Trigger input source TIMER16 BRKIN
TRIGSEL_INPUT_TIMER22_TRGO0	Trigger input source TIMER22 TRGO0
TRIGSEL_INPUT_TIMER22_CH0	Trigger input source TIMER22 CH0
TRIGSEL_INPUT_TIMER22_CH1	Trigger input source TIMER22 CH1
TRIGSEL_INPUT_TIMER22_CH2	Trigger input source TIMER22 CH2

Member name	Function description
TRIGSEL_INPUT_TIMER22_CH3	Trigger input source TIMER22 CH3
TRIGSEL_INPUT_TIMER22_ETI	Trigger input source TIMER22 ETI
TRIGSEL_INPUT_TIMER23_TRGO0	Trigger input source TIMER23 TRGO0
TRIGSEL_INPUT_TIMER23_CH0	Trigger input source TIMER23 CH0
TRIGSEL_INPUT_TIMER23_CH1	Trigger input source TIMER23 CH1
TRIGSEL_INPUT_TIMER23_CH2	Trigger input source TIMER23 CH2
TRIGSEL_INPUT_TIMER23_CH3	Trigger input source TIMER22 CH3
TRIGSEL_INPUT_TIMER23_ETI	Trigger input source TIMER23 ETI
TRIGSEL_INPUT_TIMER40_TRGO0	Trigger input source TIMER40 TRGO0
TRIGSEL_INPUT_TIMER40_CH0	Trigger input source TIMER40 CH0
TRIGSEL_INPUT_TIMER40_CH1	Trigger input source TIMER40 CH1
TRIGSEL_INPUT_TIMER40_MCH0	Trigger input source TIMER40 MCH0
TRIGSEL_INPUT_TIMER40_BRKIN	Trigger input source TIMER40 BRKIN
TRIGSEL_INPUT_TIMER41_TRGO0	Trigger input source TIMER41 TRGO0
TRIGSEL_INPUT_TIMER41_CH0	Trigger input source TIMER41 CH0
TRIGSEL_INPUT_TIMER41_CH1	Trigger input source TIMER41 CH1
TRIGSEL_INPUT_TIMER41_MCH0	Trigger input source TIMER41 MCH0
TRIGSEL_INPUT_TIMER41_BRKIN	Trigger input source TIMER41 BRKIN
TRIGSEL_INPUT_TIMER42_TRGO0	Trigger input source TIMER42 TRGO0
TRIGSEL_INPUT_TIMER42_CH0	Trigger input source TIMER42 CH0
TRIGSEL_INPUT_TIMER42_CH1	Trigger input source TIMER42 CH1
TRIGSEL_INPUT_TIMER42_MCH0	Trigger input source TIMER42 MCH0
TRIGSEL_INPUT_TIMER42_BRKIN	Trigger input source TIMER42 BRKIN
TRIGSEL_INPUT_TIMER43_TRGO0	Trigger input source TIMER43 TRGO0
TRIGSEL_INPUT_TIMER43_CH0	Trigger input source TIMER43 CH0
TRIGSEL_INPUT_TIMER43_CH1	Trigger input source TIMER43 CH1
TRIGSEL_INPUT_TIMER43_MCH0	Trigger input source TIMER43 MCH0
TRIGSEL_INPUT_TIMER43_BRKIN	Trigger input source TIMER43 BRKIN
TRIGSEL_INPUT_TIMER44_TRGO0	Trigger input source TIMER44 TRGO0
TRIGSEL_INPUT_TIMER44_CH0	Trigger input source TIMER44 CH0
TRIGSEL_INPUT_TIMER44_CH1	Trigger input source TIMER44 CH1
TRIGSEL_INPUT_TIMER44_MCH0	Trigger input source TIMER44 MCH0
TRIGSEL_INPUT_TIMER44_BRKIN	Trigger input source TIMER44 BRKIN
TRIGSEL_INPUT_TIMER50_TRGO0	Trigger input source TIMER50 TRGO0
TRIGSEL_INPUT_TIMER51_TRGO0	Trigger input source TIMER51 TRGO0
TRIGSEL_INPUT_RTC_ALARM	Trigger input source RTC alarm
TRIGSEL_INPUT_RTC_TPTS	Trigger input source RTC TPTS
TRIGSEL_INPUT_ADC0_WD0_OUT	Trigger input source ADC0 watchdog0 output
TRIGSEL_INPUT_ADC0_WD1_OUT	Trigger input source ADC0 watchdog1 output
TRIGSEL_INPUT_ADC0_WD2_OUT	Trigger input source ADC0 watchdog2 output
TRIGSEL_INPUT_ADC1_WD0_OUT	Trigger input source ADC1 watchdog0 output
TRIGSEL_INPUT_ADC1_WD1_OUT	Trigger input source ADC1 watchdog1 output

Member name	Function description
TRIGSEL_INPUT_ADC1_WD2_OUT	Trigger input source ADC1 watchdog2 output
TRIGSEL_INPUT_ADC2_WD0_OUT	Trigger input source ADC2 watchdog0 output
TRIGSEL_INPUT_ADC2_WD1_OUT	Trigger input source ADC2 watchdog1 output
TRIGSEL_INPUT_ADC2_WD2_OUT	Trigger input source ADC2 watchdog2 output
TRIGSEL_INPUT_CMP0_OUT	Trigger input source CMP0_OUT
TRIGSEL_INPUT_CMP1_OUT	Trigger input source CMP1_OUT
TRIGSEL_INPUT_SAI0_AFS_OUT	Trigger input source SAI0_AFS_OUT
TRIGSEL_INPUT_SAI0_BFS_OUT	Trigger input source SAI0_BFS_OUT
TRIGSEL_INPUT_SAI2_AFS_OUT	Trigger input source SAI2_AFS_OUT
TRIGSEL_INPUT_SAI2_BFS_OUT	Trigger input source SAI2_BFS_OUT

### Enum trigsel\_periph\_enum

**Table 3-1254. Enum trigsel\_periph\_enum**

Member name	Function description
TRIGSEL_OUTPUT_TRIGSEL_OUT0	Output target peripheral TRIGSEL_OUT0 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT1	Output target peripheral TRIGSEL_OUT1 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT2	Output target peripheral TRIGSEL_OUT2 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT3	Output target peripheral TRIGSEL_OUT3 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT4	Output target peripheral TRIGSEL_OUT4 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT5	Output target peripheral TRIGSEL_OUT5 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT6	Output target peripheral TRIGSEL_OUT6 pin
TRIGSEL_OUTPUT_TRIGSEL_OUT7	Output target peripheral TRIGSEL_OUT7 pin
TRIGSEL_OUTPUT_ADC0_REGTRG	Output target peripheral ADC0_REGTRG
TRIGSEL_OUTPUT_ADC0_INSTRG	Output target peripheral ADC0_INSTRG
TRIGSEL_OUTPUT_ADC1_REGTRG	Output target peripheral ADC1_REGTRG
TRIGSEL_OUTPUT_ADC1_INSTRG	Output target peripheral ADC1_INSTRG
TRIGSEL_OUTPUT_ADC2_REGTRG	Output target peripheral ADC2_REGTRG
TRIGSEL_OUTPUT_ADC2_INSTRG	Output target peripheral ADC2_INSTRG
TRIGSEL_OUTPUT_DAC_OUT0_EXTRG	Output target peripheral DAC_OUT0_EXTRG
TRIGSEL_OUTPUT_DAC_OUT1_EXTRG	Output target peripheral DAC_OUT1_EXTRG
TRIGSEL_OUTPUT_TIMER0_BRKIN0	Output target peripheral TIMER0_BRKIN0
TRIGSEL_OUTPUT_TIMER0_BRKIN1	Output target peripheral TIMER0_BRKIN1
TRIGSEL_OUTPUT_TIMER0_BRKIN2	Output target peripheral TIMER0_BRKIN2
TRIGSEL_OUTPUT_TIMER7_BRKIN0	Output target peripheral TIMER7_BRKIN0
TRIGSEL_OUTPUT_TIMER7_BRKIN1	Output target peripheral TIMER7_BRKIN1
TRIGSEL_OUTPUT_TIMER7_BRKIN2	Output target peripheral TIMER7_BRKIN2
TRIGSEL_OUTPUT_TIMER14_BRKIN0	Output target peripheral TIMER14_BRKIN0
TRIGSEL_OUTPUT_TIMER15_BRKIN0	Output target peripheral TIMER15_BRKIN0
TRIGSEL_OUTPUT_TIMER16_BRKIN0	Output target peripheral TIMER16_BRKIN0
TRIGSEL_OUTPUT_TIMER40_BRKIN0	Output target peripheral TIMER40_BRKIN0
TRIGSEL_OUTPUT_TIMER41_BRKIN0	Output target peripheral TIMER41_BRKIN0



Member name	Function description
TRIGSEL_OUTPUT_TIMER42_BRKIN0	Output target peripheral TIMER42_BRKIN0
TRIGSEL_OUTPUT_TIMER43_BRKIN0	Output target peripheral TIMER43_BRKIN0
TRIGSEL_OUTPUT_TIMER44_BRKIN0	Output target peripheral TIMER44_BRKIN0
TRIGSEL_OUTPUT_CAN0_EX_TIME_TICK	Output target peripheral CAN0_EX_TIME_TICK
TRIGSEL_OUTPUT_CAN1_EX_TIME_TICK	Output target peripheral CAN1_EX_TIME_TICK
TRIGSEL_OUTPUT_CAN2_EX_TIME_TICK	Output target peripheral CAN2_EX_TIME_TICK
TRIGSEL_OUTPUT_LPPTS_TRG	Output target peripheral LPPTS_TRG
TRIGSEL_OUTPUT_TIMER0_ETI	Output target peripheral TIMER0_ETI
TRIGSEL_OUTPUT_TIMER1_ETI	Output target peripheral TIMER1_ETI
TRIGSEL_OUTPUT_TIMER2_ETI	Output target peripheral TIMER2_ETI
TRIGSEL_OUTPUT_TIMER3_ETI	Output target peripheral TIMER3_ETI
TRIGSEL_OUTPUT_TIMER4_ETI	Output target peripheral TIMER4_ETI
TRIGSEL_OUTPUT_TIMER7_ETI	Output target peripheral TIMER7_ETI
TRIGSEL_OUTPUT_TIMER22_ETI	Output target peripheral TIMER22_ETI
TRIGSEL_OUTPUT_TIMER23_ETI	Output target peripheral TIMER23_ETI
TRIGSEL_OUTPUT_EDOUT_TRG	Output target peripheral EDOUT_TRG
TRIGSEL_OUTPUT_HPDF_ITR	Output target peripheral HPDF_ITR
TRIGSEL_OUTPUT_TIMER0_ITI14	Output target peripheral TIMER0_ITI14
TRIGSEL_OUTPUT_TIMER1_ITI14	Output target peripheral TIMER1_ITI14
TRIGSEL_OUTPUT_TIMER2_ITI14	Output target peripheral TIMER2_ITI14
TRIGSEL_OUTPUT_TIMER3_ITI14	Output target peripheral TIMER3_ITI14
TRIGSEL_OUTPUT_TIMER4_ITI14	Output target peripheral TIMER4_ITI14
TRIGSEL_OUTPUT_TIMER7_ITI14	Output target peripheral TIMER7_ITI14
TRIGSEL_OUTPUT_TIMER14_ITI14	Output target peripheral TIMER14_ITI14
TRIGSEL_OUTPUT_TIMER22_ITI14	Output target peripheral TIMER22_ITI14
TRIGSEL_OUTPUT_TIMER23_ITI14	Output target peripheral TIMER23_ITI14
TRIGSEL_OUTPUT_TIMER40_ITI14	Output target peripheral TIMER40_ITI14
TRIGSEL_OUTPUT_TIMER41_ITI14	Output target peripheral TIMER41_ITI14
TRIGSEL_OUTPUT_TIMER42_ITI14	Output target peripheral TIMER42_ITI14
TRIGSEL_OUTPUT_TIMER43_ITI14	Output target peripheral TIMER43_ITI14
TRIGSEL_OUTPUT_TIMER44_ITI14	Output target peripheral TIMER44_ITI14

## trigsel\_deinit

The description of trigsel\_deinit is shown as below:

**Table 3-1255. Function trigsel\_deinit**

Function name	trigsel_deinit
Function prototype	void trigsel_deinit(void);
Function descriptions	Deinitialize TRIGSEL

<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize TRIGSEL */
```

```
trigsel_deinit();
```

### trigsel\_init

The description of trigsel\_init is shown as below:

**Table 3-1256. Function trigsel\_init**

<b>Function name</b>	trigsel_init
<b>Function prototype</b>	void trigsel_init(trigsel_periph_enum target_periph, trigsel_source_enum trigger_source);
<b>Function descriptions</b>	Set the trigger input signal for target peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_periph</b>	Target peripheral value, refer to <a href="#">Table 3-1254. Enum trigsel_periph_enum</a>
<b>Input parameter{in}</b>	
<b>trigger_source</b>	Trigger source value, refer to <a href="#">Table 3-1253. Enum trigsel_source_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select TIMER0_CH2 to trigger ADC0 */
```

```
trigsel_init(TRIGSEL_OUTPUT_ADC0_REGTRG, TRIGSEL_INPUT_TIMER0_CH2);
```

### trigsel\_trigger\_source\_get

The description of trigsel\_trigger\_source\_get is shown as below:

**Table 3-1257. Function trigsel\_trigger\_source\_get**

<b>Function name</b>	trigsel_trigger_source_get
----------------------	----------------------------

<b>Function prototype</b>	trigsel_source_enum trigsel_trigger_source_get(trigsel_periph_enum target_periph);
<b>Function descriptions</b>	Get the trigger input signal for target peripheral
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_periph</b>	Target peripheral value, refer to <a href="#">Table 3-1254. Enum trigsel_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>trigger_source</b>	Trigger source value, refer to <a href="#">Table 3-1253. Enum trigsel_source_enum</a>

Example:

```
/* get the trigger input signal for ADC0 */
trigsel_source_enum input_signal;
input_signal = trigsel_trigger_source_get(TRIGSEL_OUTPUT_ADC0_REGTRG);
```

### trigsel\_register\_lock\_set

The description of trigsel\_register\_lock\_set is shown as below:

**Table 3-1258. Function trigsel\_register\_lock\_set**

<b>Function name</b>	trigsel_register_lock_set
<b>Function prototype</b>	void trigsel_register_lock_set(trigsel_periph_enum target_periph);
<b>Function descriptions</b>	Lock the trigger register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_periph</b>	Target peripheral value, refer to <a href="#">Table 3-1254. Enum trigsel_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* lock the trigger register for ADC0 */
trigsel_register_lock_set(TRIGSEL_OUTPUT_ADC0_REGTRG);
```

### trigsel\_register\_lock\_get

The description of trigsel\_register\_lock\_get is shown as below:

**Table 3-1259. Function trigsel\_register\_lock\_get**

<b>Function name</b>	trigsel_register_lock_get
<b>Function prototype</b>	FlagStatus trigsel_register_lock_get(trigsel_periph_enum target_periph);
<b>Function descriptions</b>	Get the trigger register lock status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>target_periph</b>	Target peripheral value, refer to <a href="#">Table 3-1254. Enum trigsel_periph_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the trigger register lock status of ADC0 */
```

```
FlagStatus status;
```

```
status = trigsel_register_lock_get(TRIGSEL_OUTPUT_ADC0_REGTRG);
```

## 3.35. TRNG

The true random number generator (TRNG) module can generate a 32-bit value using continuous analog noise and it has been pre-certified NIST SP800-90B. The TRNG registers are listed in chapter [3.35.1](#). the TRNG firmware functions are introduced in chapter [3.35.2](#).

### 3.35.1. Descriptions of Peripheral registers

TRNG registers are listed in the table shown as below:

**Table 3-1260. TRNG Registers**

Registers	Descriptions
TRNG_CTL	control register
TRNG_STAT	status register
TRNG_DATA	data register
TRNG_HTCFG	health tests configure register

### 3.35.2. Descriptions of Peripheral functions

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

**Table 3-1261. TRNG firmware function**

Function name	Function description
trng_deinit	deinitialize the TRNG

Function name	Function description
trng_enable	enable the TRNG interface
trng_disable	disable the TRNG interface
trng_lock	lock the TRNG control bits
trng_mode_config	Configure TRNG working mode
trng_postprocessing_enable	enable the TRNG post-processing module
trng_postprocessing_disable	disable the TRNG post-processing module
trng_conditioning_enable	enable the TRNG conditioning module
trng_conditioning_disable	disable the TRNG conditioning module
trng_conditioning_input_bitwidth	configure TRNG conditioning module input bitwidth
trng_conditioning_output_bitwidth	configure TRNG conditioning module output bitwidth
trng_replace_test_enable	enable TRNG replace test
trng_replace_test_disable	disable TRNG replace test
trng_hash_init_enable	enable hash algorithm init when conditioning module enabled
trng_hash_init_disable	disable hash algorithm init when conditioning module enabled
trng_powermode_config	configure TRNG analog power mode
trng_clockdiv_config	configure TRNG clock divider
trng_clockerror_detection_enable	enable the TRNG clock error detection
trng_clockerror_detection_disable	disable the TRNG clock error detection
trng_get_true_random_data	get the true random data
trng_conditioning_reset_enable	enable the conditioning logic reset
trng_conditioning_reset_disable	disable the conditioning logic reset
trng_conditioning_algo_config	configure the conditioning module hash algorithm
trng_health_tests_config	configure health tests default value
trng_flag_get	get the TRNG status flags
trng_interrupt_enable	the TRNG interrupt enable
trng_interrupt_disable	the TRNG interrupt disable
trng_interrupt_flag_get	get the TRNG interrupt flags
trng_interrupt_flag_clear	clear the TRNG interrupt flags

## Enum trng\_inmod\_enum

Table 3-1262. Enum trng\_inmod\_enum

Member name	Function description
TRNG_INMOD_256BIT	conditioning module input bitwidth 256bits
TRNG_INMOD_440BIT	conditioning module input bitwidth 440bits

## Enum trng\_outmod\_enum

Table 3-1263. Enum trng\_outmod\_enum

Member name	Function description
TRNG_OUTMOD_128BIT	conditioning module output bitwidth 128bits
TRNG_OUTMOD_256BIT	conditioning module output bitwidth 256bits

## Enum trng\_modsel\_enum

**Table 3-1264. Enum trng\_modsel\_enum**

Member name	Function description
TRNG_MODSEL_LFSR	TRNG working in LFSR mode
TRNG_MODSEL_NIST	TRNG working in NIST mode

## Enum trng\_flag\_enum

**Table 3-1265. Enum trng\_flag\_enum**

Member name	Function description
TRNG_FLAG_DRDY	random data ready status
TRNG_FLAG_CECS	clock error current status
TRNG_FLAG_SECS	seed error current status

## Enum trng\_int\_flag\_enum

**Table 3-1266. Enum trng\_int\_flag\_enum**

Member name	Function description
TRNG_INT_FLAG_CEIF	clock error interrupt flag
TRNG_INT_FLAG_SEIF	seed error interrupt flag

## trng\_deinit

The description of trng\_deinit is shown as below:

**Table 3-1267. Function trng\_deinit**

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

Example:

```
/* TRNG deinit */
```

```
trng_deinit( );
```

## trng\_enable

The description of trng\_enable is shown as below:

**Table 3-1268. Function trng\_enable**

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

Example:

```
/* enable the TRNG interface */
trng_enable( );
```

## trng\_disable

The description of trng\_disable is shown as below:

**Table 3-1269. Function trng\_disable**

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

Example:

```
/* disable the TRNG interface */
trng_disable( );
```

## trng\_lock

The description of trng\_lock is shown as below:

**Table 3-1270. Function trng\_lock**

<b>Function name</b>	trng_lock
<b>Function prototype</b>	void trng_lock (void);
<b>Function descriptions</b>	lock the TRNG control bits
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* lock the TRNG control bits */

trng_lock( );
```

### trng\_mode\_config

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

**Table 3-1271. Function trng\_mode\_config**

<b>Function name</b>	trng_mode_config
<b>Function prototype</b>	void trng_mode_config(trng_modsel_enum mode_select);
<b>Function descriptions</b>	configure TRNG working mode
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>mode_select</b>	-
TRNG_MODSEL_LFSR	TRNG working in LFSR mode
TRNG_MODSEL_NIST	TRNG working in NIST mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();
```



```

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_postprocessing\_enable

The description of trng\_postprocessing\_enable is shown as below:

**Table 3-1272. Function trng\_postprocessing\_enable**

<b>Function name</b>	trng_postprocessing_enable
<b>Function prototype</b>	void trng_postprocessing_enable(void);
<b>Function descriptions</b>	enable the TRNG post-processing module
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

## trng\_postprocessing\_disable

The description of trng\_postprocessing\_disable is shown as below:

**Table 3-1273. Function trng\_postprocessing\_disable**

<b>Function name</b>	trng_postprocessing_disable
<b>Function prototype</b>	void trng_postprocessing_disable(void);
<b>Function descriptions</b>	disable the TRNG post-processing module
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_disable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

## trng\_conditioning\_enable

The description of trng\_conditioning\_enable is shown as below:

**Table 3-1274. Function trng\_conditioning\_enable**

<b>Function name</b>	trng_conditioning_enable
<b>Function prototype</b>	void trng_conditioning_enable(void);
<b>Function descriptions</b>	enable the TRNG conditioning module
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-

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

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_conditioning\_disable

The description of trng\_conditioning\_disable is shown as below:

**Table 3-1275. Function trng\_conditioning\_disable**

Function name	trng_conditioning_disable
Function prototype	void trng_conditioning_disable(void);
Function descriptions	disable the TRNG conditioning module
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```

/* configure TRNG module */

```

```

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_disable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_conditioning\_input\_bitwidth

The description of trng\_conditioning\_input\_bitwidth is shown as below:

**Table 3-1276. Function trng\_disable**

<b>Function name</b>	trng_conditioning_input_bitwidth
<b>Function prototype</b>	void trng_conditioning_input_bitwidth(trng_inmod_enum input_bitwidth);
<b>Function descriptions</b>	configure TRNG conditioning module input bitwidth
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>input_bitwidth</b>	refer to enum <a href="#">Table 3-1262. Enum trng_inmod_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

```

```
trng_clockerror_detection_enable();
```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_conditioning\_output\_bitwidth

The description of trng\_conditioning\_output\_bitwidth is shown as below:

**Table 3-1277. Function trng\_conditioning\_output\_bitwidth**

<b>Function name</b>	trng_conditioning_output_bitwidth
<b>Function prototype</b>	void trng_conditioning_output_bitwidth(trng_outmod_enum output_bitwidth);
<b>Function descriptions</b>	configure TRNG conditioning module output bitwidth
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>output_bitwidth</b>	refer to enum <a href="#">Table 3-1263. Enum trng_outmod_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();
```

### trng\_replace\_test\_enable

The description of trng\_replace\_test\_enable is shown as below:

**Table 3-1278. Function trng\_replace\_test\_enable**

<b>Function name</b>	trng_replace_test_enable
----------------------	--------------------------

<b>Function prototype</b>	void trng_replace_test_enable(void);
<b>Function descriptions</b>	enable TRNG replace test
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* enable TRNG replace test */

trng_deinit();

trng_conditioning_reset_enable();

trng_replace_test_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_replace\_test\_disable

The description of trng\_replace\_test\_disable is shown as below:

**Table 3-1279. Function trng\_replace\_test\_disable**

<b>Function name</b>	trng_replace_test_disable
<b>Function prototype</b>	void trng_replace_test_disable(void);
<b>Function descriptions</b>	disable TRNG replace test
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* disable TRNG replace test */

trng_deinit();

trng_conditioning_reset_enable();

trng_replace_test_disable();

```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_hash\_init\_enable

The description of trng\_hash\_init\_enable is shown as below:

**Table 3-1280. Function trng\_hash\_init\_enable**

<b>Function name</b>	trng_hash_init_enable
<b>Function prototype</b>	void trng_hash_init_enable(void);
<b>Function descriptions</b>	enable hash algorithm init when conditioning module enabled
<b>Precondition</b>	trng_conditioning_reset_enable / trng_conditioning_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable hash algorithm init when conditioning module enabled */
```

```
trng_deinit();
```

```
trng_conditioning_reset_enable();
```

```
trng_conditioning_enable();
```

```
trng_hash_init_enable();
```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_hash\_init\_disable

The description of trng\_hash\_init\_disable is shown as below:

**Table 3-1281. Function trng\_hash\_init\_disable**

<b>Function name</b>	trng_hash_init_disable
<b>Function prototype</b>	void trng_hash_init_disable(void);
<b>Function descriptions</b>	disable hash algorithm init when conditioning module enabled
<b>Precondition</b>	trng_conditioning_reset_enable / trng_conditioning_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	

-	-
Return value	
-	-

Example:

```

/* disable the TRNG interface */

trng_deinit();

trng_conditioning_reset_enable();

trng_conditioning_enable();

trng_hash_init_disable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_powermode\_config

The description of trng\_powermode\_config is shown as below:

**Table 3-1282. Function trng\_powermode\_config**

<b>Function name</b>	trng_powermode_config
<b>Function prototype</b>	void trng_powermode_config(uint32_t powermode);
<b>Function descriptions</b>	configure TRNG analog power mode
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>powermode</b>	the power mode selection
<i>TRNG_NR_ULATRLOW</i>	TRNG analog power mode ulatrlow
<i>TRNG_NR_LOW</i>	TRNG analog power mode low
<i>TRNG_NR_MEDIUM</i>	TRNG analog power mode medium
<i>TRNG_NR_HIGH</i>	TRNG analog power mode high
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure the TRNG analog power mode as high */

trng_deinit();

trng_conditioning_reset_enable();

trng_powermode_config(TRNG_NR_HIGH);

```



```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_clockdiv\_config

The description of trng\_clockdiv\_config is shown as below:

**Table 3-1283. Function trng\_disable**

<b>Function name</b>	trng_clockdiv_config
<b>Function prototype</b>	void trng_clockdiv_config(uint32_t clkdiv);
<b>Function descriptions</b>	configure TRNG clock divider
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>clkdiv</b>	the TRNG clock divider
<i>TRNG_CLK_DIVx</i>	clock division coefficient x, x = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the TRNG clock frequency division coefficient to 64 */
```

```
trng_deinit();
```

```
trng_conditioning_reset_enable();
```

```
trng_clockdiv_config(TRNG_CLK_DIV64);
```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_clockerror\_detection\_enable

The description of trng\_clockerror\_detection\_enable is shown as below:

**Table 3-1284. Function trng\_clockerror\_detection\_enable**

<b>Function name</b>	trng_clockerror_detection_enable
<b>Function prototype</b>	void trng_clockerror_detection_enable(void);
<b>Function descriptions</b>	enable the TRNG clock error detection
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable the TRNG clock error detection */
```

```
trng_deinit();
```

```
trng_conditioning_reset_enable();
```

```
trng_clockerror_detection_enable();
```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

### trng\_clockerror\_detection\_disable

The description of trng\_clockerror\_detection\_disable is shown as below:

**Table 3-1285. Function trng\_clockerror\_detection\_disable**

Function name	trng_clockerror_detection_disable
Function prototype	void trng_clockerror_detection_disable(void);
Function descriptions	disable the TRNG clock error detection
Precondition	trng_conditioning_reset_enable
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable the TRNG clock error detection */
```

```
trng_deinit();
```

```
trng_conditioning_reset_enable();
```

```
trng_clockerror_detection_disable();
```

```
trng_enable();
```

```
trng_conditioning_reset_disable();
```

## trng\_get\_true\_random\_data

The description of trng\_get\_true\_random\_data is shown as below:

**Table 3-1286. Function trng\_get\_true\_random\_data**

<b>Function name</b>	trng_get_true_random_data
<b>Function prototype</b>	uint32_t trng_get_true_random_data(void);
<b>Function descriptions</b>	get the true random data
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	0x0 – 0xFFFFFFFF

Example:

```
/* get the true random data */
uint32_t data = 0;
data = trng_get_true_random_data();
```

## trng\_conditioning\_reset\_enable

The description of trng\_conditioning\_reset\_enable is shown as below:

**Table 3-1287. Function trng\_conditioning\_reset\_enable**

<b>Function name</b>	trng_conditioning_reset_enable
<b>Function prototype</b>	void trng_conditioning_reset_enable (void)
<b>Function descriptions</b>	enable the conditioning logic reset
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TRNG module */
trng_deinit();
trng_conditioning_reset_enable();
```

```

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

trng_conditioning_reset_disable();

```

### trng\_conditioning\_reset\_disable

The description of trng\_conditioning\_reset\_disable is shown as below:

**Table 3-1288. Function trng\_conditioning\_reset\_disable**

<b>Function name</b>	trng_conditioning_reset_disable
<b>Function prototype</b>	void trng_conditioning_reset_disable(void)
<b>Function descriptions</b>	disable the conditioning logic reset
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_enable();

```

```
trng_conditioning_reset_disable();
```

### trng\_conditioning\_algo\_config

The description of trng\_conditioning\_algo\_config is shown as below:

**Table 3-1289. Function trng\_conditioning\_algo\_config**

<b>Function name</b>	trng_conditioning_algo_config
<b>Function prototype</b>	void trng_conditioning_algo_config(uint32_t module_algo)
<b>Function descriptions</b>	configure the conditioning module hash algorithm
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>module_algo</b>	module hash algorithm
TRNG_ALGO_SHA1	TRNG conditioning module hash SHA1
TRNG_ALGO_MD5	TRNG conditioning module hash MD5
TRNG_ALGO_SHA224	TRNG conditioning module hash SHA224
TRNG_ALGO_SHA256	TRNG conditioning module hash SHA256
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_conditioning_algo_config(TRNG_ALGO_SHA1);

trng_enable();

trng_conditioning_reset_disable();
```

### trng\_health\_tests\_config

The description of trng\_health\_tests\_config is shown as below:

**Table 3-1290. Function trng\_health\_tests\_config**

<b>Function name</b>	trng_health_tests_config
<b>Function prototype</b>	void trng_health_tests_config(uint32_t adpo_threshold, uint8_t rep_threshold)
<b>Function descriptions</b>	configure health tests default value
<b>Precondition</b>	trng_conditioning_reset_enable
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>adpo_threshold</b>	adaptive proportion test threshold value
<b>Input parameter{in}</b>	
<b>rep_threshold</b>	repetitive (00/11) test threshold value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* configure TRNG module */

trng_deinit();

trng_conditioning_reset_enable();

trng_mode_config(TRNG_MODSEL_NIST);

trng_postprocessing_enable();

trng_conditioning_enable();

trng_conditioning_input_bitwidth(TRNG_INMOD_440BIT);

trng_conditioning_output_bitwidth(TRNG_OUTMOD_256BIT);

trng_clockerror_detection_enable();

trng_health_tests_config(700, 50);

trng_enable();

```

### trng\_flag\_get

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

**Table 3-1291. Function trng\_flag\_get**

<b>Function name</b>	trng_flag_get
<b>Function prototype</b>	FlagStatus trng_flag_get(trng_flag_enum flag);
<b>Function descriptions</b>	get the trng status flags
<b>Precondition</b>	-
<b>The called functions</b>	-

Input parameter{in}	
flag	trng status flag, refer to <a href="#">Table 3-1265. Enum trng_flag_enum</a>
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the trng status flags */

FlagStatus status;

status = trng_flag_get(TRNG_FLAG_DRDY);
```

### trng\_interrupt\_enable

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

**Table 3-1292. Function trng\_interrupt\_enable**

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

Example:

```
/* enable the TRNG interrupt */

trng_interrupt_enable( );
```

### trng\_interrupt\_disable

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

**Table 3-1293. Function trng\_interrupt\_disable**

Function name	trng_interrupt_disable
Function prototype	void trng_interrupt_disable(void);
Function descriptions	disable the TRNG interrupt
Precondition	-
The called functions	-
Input parameter{in}	

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

Example:

```
/* disable the TRNG interrupt */
```

```
trng_interrupt_disable( );
```

### trng\_interrupt\_flag\_get

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

**Table 3-1294. Function trng\_interrupt\_flag\_get**

<b>Function name</b>	trng_interrupt_flag_get
<b>Function prototype</b>	FlagStatus trng_interrupt_flag_get(trng_int_flag_enum int_flag)
<b>Function descriptions</b>	get the trng interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>int_flag</b>	trng interrupt flag, refer to <a href="#">Table 3-1266. Enum trng_int_flag_enum</a>
Output parameter{out}	
-	-
Return value	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get the trng interrupt flag */
```

```
FlagStatus status = RESET;
```

```
status = trng_interrupt_flag_get(TRNG_INT_FLAG_CEIF);
```

### trng\_interrupt\_flag\_clear

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

**Table 3-1295. Function trng\_interrupt\_flag\_clear**

<b>Function name</b>	trng_interrupt_flag_clear
<b>Function prototype</b>	void trng_interrupt_flag_clear(trng_int_flag_enum int_flag)
<b>Function descriptions</b>	clear the trng interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>int_flag</b>	trng interrupt flag, refer to <a href="#">Table 3-1266. Enum trng_int_flag_enum</a>



Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*clear the trng interrupt flag */
```

```
trng_interrupt_flag_clear(TRNG_INT_FLAG_CEIF);
```

## 3.36. USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. The USART registers are listed in chapter [3.36.1](#), the USART firmware functions are introduced in chapter [3.36.2](#).

### 3.36.1. Descriptions of Peripheral registers

USART registers are listed in the table shown as below:

**Table 3-1296. USART Registers**

Registers	Descriptions
USART_CTL0	control register 0
USART_CTL1	control register 1
USART_CTL2	control register 2
USART_BAUD	baud rate generator register
USART_GP	prescaler and guard time configuration register
USART_RT	receiver timeout register
USART_CMD	command register
USART_STAT	status register
USART_INTC	interrupt status clear register
USART_RDATA	receive data register
USART_TDATA	transmit data register
USART_CHC	coherence control register
USART_FCS	FIFO control and status register

### 3.36.2. Descriptions of Peripheral functions

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

**Table 3-1297. USART firmware function**

Function name	Function description
usart_deinit	reset USART
usart_baudrate_set	configure USART baud rate value

Function name	Function description
usart_parity_config	configure USART parity function
usart_word_length_set	configure USART word length
usart_stop_bit_set	configure USART stop bit length
usart_enable	enable USART
usart_disable	disable USART
usart_transmit_config	configure USART transmitter
usart_receive_config	configure USART receiver
usart_data_first_config	data is transmitted/received with the LSB/MSB first
usart_invert_config	configure USART inverted
usart_overrun_enable	enable the USART overrun function
usart_overrun_disable	disable the USART overrun function
usart_oversample_config	configure the USART oversample mode
usart_sample_bit_config	configure sample bit method
usart_receiver_timeout_enable	enable receiver timeout
usart_receiver_timeout_disable	disable receiver timeout
usart_receiver_timeout_threshold_config	configure receiver timeout threshold
usart_data_transmit	USART transmit data function
usart_data_receive	USART receive data function
usart_command_enable	enable USART command
usart_address_0_match_mode_enable	enable address 0 match mode
usart_address_0_match_mode_disable	disable address 0 match mode
usart_address_1_match_mode_enable	enable address 1 match mode
usart_address_1_match_mode_disable	disable address 1 match mode
usart_address_0_config	configure address 0 of the USART
usart_address_1_config	configure address 1 of the USART
usart_address_0_detection_mode_config	configure address 0 detection mode
usart_address_1_detection_mode_config	configure address 1 detection mode
usart_mute_mode_enable	enable mute mode
usart_mute_mode_disable	disable mute mode
usart_mute_mode_wakeup_config	configure wakeup method in mute mode
usart_lin_mode_enable	enable LIN mode
usart_lin_mode_disable	disable LIN mode
usart_lin_break_detection_length_config	LIN break detection length
usart_halfduplex_enable	enable half-duplex mode

Function name	Function description
usart_halfduplex_disable	disable half-duplex mode
usart_clock_enable	enable clock
usart_clock_disable	disable clock
usart_synchronous_clock_config	configure USART synchronous mode parameters
usart_guard_time_config	configure guard time value in smartcard mode
usart_smartcard_mode_enable	enable smartcard mode
usart_smartcard_mode_disable	disable smartcard mode
usart_smartcard_mode_nack_enable	enable NACK in smartcard mode
usart_smartcard_mode_nack_disable	disable NACK in smartcard mode
usart_smartcard_mode_early_nack_enable	enable early NACK in smartcard mode
usart_smartcard_mode_early_nack_disable	disable early NACK in smartcard mode
usart_smartcard_autoretry_config	configure smartcard auto-retry number
usart_block_length_config	configure block length
usart_irda_mode_enable	enable IrDA mode
usart_irda_mode_disable	disable IrDA mode
usart_prescaler_config	configure the peripheral clock prescaler in USART IrDA low-power or smartcard mode
usart_irda_lowpower_config	configure IrDA low-power
usart_hardware_flow_rts_config	configure hardware flow control RTS
usart_hardware_flow_cts_config	configure hardware flow control CTS
usart_hardware_flow_coherence_config	configure hardware flow control coherence mode
usart_rs485_driver_enable	enable RS485 driver
usart_rs485_driver_disable	disable RS485 driver
usart_driver_asserttime_config	configure driver enable assertion time
usart_driver_deasserttime_config	configure driver enable de-assertion time
usart_depolarity_config	configure driver enable polarity mode
usart_dma_receive_config	configure USART DMA for reception
usart_dma_transmit_config	configure USART DMA for transmission
usart_reception_error_dma_disable	disable DMA on reception error
usart_reception_error_dma_enable	enable DMA on reception error
usart_wakeup_enable	enable USART to wakeup the mcu from deep-sleep mode
usart_wakeup_disable	disable USART to wakeup the mcu from deep-sleep mode
usart_wakeup_mode_config	configure the USART wakeup mode from deep-sleep mode
usart_fifo_enable	enable FIFO
usart_fifo_disable	disable FIFO
usart_transmit_fifo_threshold_config	configure transmit FIFO threshold
usart_receive_fifo_threshold_config	configure receive FIFO threshold
usart_receive_fifo_counter_number	read receive FIFO counter number
usart_flag_get	get USART status

Function name	Function description
usart_flag_clear	clear USART status
usart_interrupt_enable	enable USART interrupt
usart_interrupt_disable	disable USART interrupt
usart_interrupt_flag_get	get USART interrupt and flag status
usart_interrupt_flag_clear	clear USART interrupt flag

## Enum usart\_flag\_enum

**Table 3-1298. Enum usart\_flag\_enum**

Member name	Function description
USART_FLAG_REA	receive enable acknowledge flag
USART_FLAG_TEA	transmit enable acknowledge flag
USART_FLAG_WU	wakeup from deep-sleep mode flag
USART_FLAG_RWU	receiver wakeup from mute mode
USART_FLAG_SB	send break flag
USART_FLAG_AM0	ADDR0 match flag
USART_FLAG_BSY	busy flag
USART_FLAG_AM1	ADDR1 match flag
USART_FLAG_EB	end of block flag
USART_FLAG_RT	receiver timeout flag
USART_FLAG_CTS	CTS level
USART_FLAG_CTSF	CTS change flag
USART_FLAG_LBD	LIN break detected flag
USART_FLAG_TBE	transmit data buffer empty
USART_FLAG_TFN	transmit FIFO not full
USART_FLAG_TC	transmission complete
USART_FLAG_RBNE	read data buffer not empty
USART_FLAG_RFN	receive FIFO not empty
USART_FLAG_IDLE	IDLE line detected flag
USART_FLAG_ORERR	overrun error
USART_FLAG_NERR	noise error flag
USART_FLAG_FERR	frame error flag
USART_FLAG_PERR	parity error flag
USART_FLAG_EPERR	early parity error flag
USART_FLAG_RFF	receive FIFO full flag
USART_FLAG_RFE	receive FIFO empty flag
USART_FLAG_TFF	transmit FIFO full flag
USART_FLAG_TFE	transmit FIFO empty flag
USART_FLAG_TFT	transmit FIFO threshold reach flag
USART_FLAG_RFT	receive FIFO threshold reach flag

## Enum usart\_interrupt\_flag\_enum

**Table 3-1299. Enum usart\_interrupt\_flag\_enum**

Member name	Function description
USART_INT_FLAG_AM1	address 1 match interrupt and flag
USART_INT_FLAG_EB	end of block interrupt and flag
USART_INT_FLAG_RT	receiver timeout interrupt and flag
USART_INT_FLAG_AM0	address 0 match interrupt and flag
USART_INT_FLAG_PERR	parity error interrupt and flag
USART_INT_FLAG_TBE	transmitter buffer empty interrupt and flag
USART_INT_FLAG_TFNF	transmit FIFO not full interrupt and flag
USART_INT_FLAG_TC	transmission complete interrupt and flag
USART_INT_FLAG_RBNE	read data buffer not empty interrupt and flag
USART_INT_FLAG_RFNE	receive FIFO not empty interrupt and flag
USART_INT_FLAG_RBNE_ORE RR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_IDLE	IDLE line detected interrupt and flag
USART_INT_FLAG_LBD	LIN break detected interrupt and flag
USART_INT_FLAG_WU	wakeup from deep-sleep mode interrupt and flag
USART_INT_FLAG_CTS	CTS interrupt and flag
USART_INT_FLAG_ERR_NERR	error interrupt and noise error flag
USART_INT_FLAG_ERR_ORER R	error interrupt and overrun error
USART_INT_FLAG_ERR_FERR	error interrupt and frame error flag
USART_INT_FLAG_TFT	transmit FIFO threshold reach interrupt and flag
USART_INT_FLAG_TFE	transmit FIFO empty interrupt and flag
USART_INT_FLAG_RFT	receive FIFO threshold reach interrupt and flag
USART_INT_FLAG_RFF	receive FIFO full interrupt and flag

## Enum usart\_interrupt\_enum

**Table 3-1300. Enum usart\_interrupt\_enum**

Member name	Function description
USART_INT_AM1	address 1 match interrupt
USART_INT_EB	end of block interrupt
USART_INT_RT	receiver timeout interrupt
USART_INT_AM0	Address 0 match interrupt
USART_INT_PERR	parity error interrupt
USART_INT_TBE	transmitter buffer empty interrupt
USART_INT_TFNF	transmit FIFO not full interrupt
USART_INT_TC	transmission complete interrupt
USART_INT_RBNE	read data buffer not empty interrupt and overrun error interrupt
USART_INT_RFNE	receive FIFO not empty interrupt and overrun error interrupt
USART_INT_IDLE	IDLE line detected interrupt

Member name	Function description
USART_INT_LBD	LIN break detected interrupt
USART_INT_WU	wakeup from deep-sleep mode interrupt
USART_INT_CTS	CTS interrupt
USART_INT_ERR	error interrupt
USART_INT_TFE	transmit FIFO empty interrupt
USART_INT_TFT	transmit FIFO threshold interrupt
USART_INT_RFT	receive FIFO threshold interrupt
USART_INT_RFF	receive FIFO full interrupt

## Enum usart\_invert\_enum

**Table 3-1301. Enum usart\_invert\_enum**

Member name	Function description
USART_DINV_ENABLE	data bit level inversion
USART_DINV_DISABLE	data bit level not inversion
USART_TXPIN_ENABLE	TX pin level inversion
USART_TXPIN_DISABLE	TX pin level not inversion
USART_RXPIN_ENABLE	RX pin level inversion
USART_RXPIN_DISABLE	RX pin level not inversion
USART_SWAP_ENABLE	swap TX/RX pins
USART_SWAP_DISABLE	not swap TX/RX pins

## usart\_deinit

The description of usart\_deinit is shown as below:

**Table 3-1302. Function usart\_deinit**

<b>Function name</b>	usart_deinit
<b>Function prototype</b>	void usart_deinit(uint32_t usart_periph);
<b>Function descriptions</b>	reset USART
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* reset USART0 */
```

```
usart_deinit(USART0);
```

## usart\_baudrate\_set

The description of usart\_baudrate\_set is shown as below:

**Table 3-1303. Function usart\_baudrate\_set**

<b>Function name</b>	usart_baudrate_set
<b>Function prototype</b>	void usart_baudrate_set(uint32_t usart_periph, uint32_t baudval);
<b>Function descriptions</b>	configure USART baud rate value
<b>Precondition</b>	-
<b>The called functions</b>	rcu_clock_freq_get
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>baudval</b>	baud rate value
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 baud rate value */
```

```
usart_baudrate_set(USART0, 115200);
```

## usart\_parity\_config

The description of usart\_parity\_config is shown as below:

**Table 3-1304. Function usart\_parity\_config**

<b>Function name</b>	usart_parity_config
<b>Function prototype</b>	void usart_parity_config(uint32_t usart_periph, uint32_t paritycfg);
<b>Function descriptions</b>	configure USART parity
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>paritycfg</b>	configure USART parity
USART_PM_NONE	no parity
USART_PM_ODD	odd parity

<i>USART_PM_EVEN</i>	even parity
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 parity */
usart_parity_config(USART0, USART_PM_EVEN);
```

### usart\_word\_length\_set

The description of usart\_word\_length\_set is shown as below:

**Table 3-1305. Function usart\_word\_length\_set**

<b>Function name</b>	usart_word_length_set
<b>Function prototype</b>	void usart_word_length_set(uint32_t usart_periph, uint32_t wlen);
<b>Function descriptions</b>	configure USART word length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>wlen</b>	USART word length configure
<i>USART_WL_8BIT</i>	8 bits
<i>USART_WL_9BIT</i>	9 bits
<i>USART_WL_7BIT</i>	7 bits
<i>USART_WL_10BIT</i>	10 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 word length */
usart_word_length_set(USART0, USART_WL_9BIT);
```

### usart\_stop\_bit\_set

The description of usart\_stop\_bit\_set is shown as below:



Table 3-1306. Function usart\_stop\_bit\_set

Function name	usart_stop_bit_set
Function prototype	void usart_stop_bit_set(uint32_t usart_periph, uint32_t stblen);
Function descriptions	configure USART stop bit length
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	
stblen	USART stop bit configure
USART_STB_1BIT	1 bit
USART_STB_0_5BIT	0.5 bit
USART_STB_2BIT	2 bits
USART_STB_1_5BIT	1.5 bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 stop bit length */
```

```
usart_stop_bit_set(USART0, USART_STB_1_5BIT);
```

### usart\_enable

The description of usart\_enable is shown as below:

Table 3-1307. Function usart\_enable

Function name	usart_enable
Function prototype	void usart_enable(uint32_t usart_periph);
Function descriptions	enable USART
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 */
usart_enable(USART0);
```

### usart\_disable

The description of usart\_disable is shown as below:

**Table 3-1308. Function usart\_disable**

<b>Function name</b>	usart_disable
<b>Function prototype</b>	void usart_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable USART
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 */
usart_disable(USART0);
```

### usart\_transmit\_config

The description of usart\_transmit\_config is shown as below:

**Table 3-1309. Function usart\_transmit\_config**

<b>Function name</b>	usart_transmit_config
<b>Function prototype</b>	void usart_transmit_config(uint32_t usart_periph, uint32_t txconfig);
<b>Function descriptions</b>	configure USART transmitter
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>txconfig</b>	enable or disable USART transmitter
USART_TRANSMIT_E	enable USART transmission

<i>NABLE</i>	
<i>USART_TRANSMIT_DISABLE</i>	disable USART transmission
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 transmitter */
```

```
usart_transmit_config(USART0,USART_TRANSMIT_ENABLE);
```

### usart\_receive\_config

The description of usart\_receive\_config is shown as below:

**Table 3-1310. Function usart\_receive\_config**

<b>Function name</b>	usart_receive_config
<b>Function prototype</b>	void usart_receive_config(uint32_t usart_periph, uint32_t rxconfig);
<b>Function descriptions</b>	configure USART receiver
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4,6, 7
<b>Input parameter{in}</b>	
<b>rxconfig</b>	enable or disable USART receiver
<i>USART_RECEIVE_ENABLE</i>	enable USART reception
<i>USART_RECEIVE_DISABLE</i>	disable USART reception
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 receiver */
```

```
usart_receive_config(USART0, USART_RECEIVE_ENABLE);
```

### usart\_data\_first\_config

The description of usart\_data\_first\_config is shown as below:

Table 3-1311. Function `usart_data_first_config`

Function name	<code>usart_data_first_config</code>
Function prototype	<code>void usart_data_first_config(uint32_t usart_periph, uint32_t msbf);</code>
Function descriptions	data is transmitted/received with the LSB/MSB first
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	USARTx peripheral
<code>USARTx</code>	x = 0, 1, 2, 5
<code>UARTx</code>	x = 3, 4, 6, 7
Input parameter{in}	
<code>msbf</code>	LSB/MSB
<code>USART_MSBF_LSB</code>	LSB first
<code>USART_MSBF_MSB</code>	MSB first
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure LSB of data first */
```

```
usart_data_first_config(USART0, USART_MSBF_LSB);
```

### `usart_invert_config`

The description of `usart_invert_config` is shown as below:

Table 3-1312. Function `usart_invert_config`

Function name	<code>usart_invert_config</code>
Function prototype	<code>void usart_invert_config(uint32_t usart_periph, usart_invert_enum invertpara);</code>
Function descriptions	configure USART inverted
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	USARTx peripheral
<code>USARTx</code>	x = 0, 1, 2, 5
<code>UARTx</code>	x = 3, 4, 6, 7
Input parameter{in}	
<code>invertpara</code>	refer to <a href="#">Table 3-1301. Enum usart_invert_enum</a>
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 inversion */
usart_invert_config(USART0, USART_DINV_ENABLE);
```

### usart\_overshoot\_enable

The description of usart\_overshoot\_enable is shown as below:

**Table 3-1313. Function usart\_overshoot\_enable**

<b>Function name</b>	usart_overshoot_enable
<b>Function prototype</b>	void usart_overshoot_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable the USART overshoot function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 overshoot */
usart_overshoot_enable(USART0);
```

### usart\_overshoot\_disable

The description of usart\_overshoot\_disable is shown as below:

**Table 3-1314. Function usart\_overshoot\_disable**

<b>Function name</b>	usart_overshoot_disable
<b>Function prototype</b>	void usart_overshoot_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable the USART overshoot function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable USART0 overrun */
usart_overrun_disable(USART0);
```

### usart\_oversample\_config

The description of usart\_oversample\_config is shown as below:

**Table 3-1315. Function usart\_oversample\_config**

<b>Function name</b>	usart_oversample_config
<b>Function prototype</b>	void usart_oversample_config(uint32_t usart_periph, uint32_t oversamp);
<b>Function descriptions</b>	configure the USART oversample mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>oversamp</b>	oversample value
USART_OVSMOD_8	oversampling by 8
USART_OVSMOD_16	oversampling by 16
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 oversampling by 8 */
usart_oversample_config(USART0, USART_OVSMOD_8);
```

### usart\_sample\_bit\_config

The description of usart\_sample\_bit\_config is shown as below:

**Table 3-1316. Function usart\_sample\_bit\_config**

<b>Function name</b>	usart_sample_bit_config
<b>Function prototype</b>	void usart_sample_bit_config(uint32_t usart_periph, uint32_t osb);
<b>Function descriptions</b>	configure the sample bit method
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>osb</b>	sample bit
<i>USART_OSB_1BIT</i>	1 bit
<i>USART_OSB_3BIT</i>	3 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config USART0 1 bit sample mode */
usart_sample_bit_config(USART0, USART_OSB_1BIT);
```

### usart\_receiver\_timeout\_enable

The description of usart\_receiver\_timeout\_enable is shown as below:

**Table 3-1317. Function usart\_receiver\_timeout\_enable**

<b>Function name</b>	usart_receiver_timeout_enable
<b>Function prototype</b>	void usart_receiver_timeout_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable receiver timeout
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 receiver timeout */
usart_receiver_timeout_enable(USART0);
```

### usart\_receiver\_timeout\_disable

The description of usart\_receiver\_timeout\_disable is shown as below:

**Table 3-1318. Function usart\_receiver\_timeout\_disable**

<b>Function name</b>	usart_receiver_timeout_disable
----------------------	--------------------------------

<b>Function prototype</b>	void usart_receiver_timeout_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable receiver timeout
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<b>USARTx</b>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 receiver timeout */
usart_receiver_timeout_disable(USART0);
```

### usart\_receiver\_timeout\_threshold\_config

The description of usart\_receiver\_timeout\_threshold\_config is shown as below:

**Table 3-1319. Function usart\_receiver\_timeout\_threshold\_config**

<b>Function name</b>	usart_receiver_timeout_threshold_config
<b>Function prototype</b>	void usart_receiver_timeout_threshold_config(uint32_t usart_periph, uint32_t timeout);
<b>Function descriptions</b>	configure receiver timeout threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<b>USARTx</b>	x = 0, 1, 2, 5
<b>Input parameter{in}</b>	
<b>rtimeout</b>	receiver timeout (0x00000000-0x00FFFFFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the receiver timeout threshold of USART0 */
usart_receiver_timeout_threshold_config(USART0, 115200*3);
```

### usart\_data\_transmit

The description of usart\_data\_transmit is shown as below:



**Table 3-1320. Function usart\_data\_transmit**

<b>Function name</b>	usart_data_transmit
<b>Function prototype</b>	void usart_data_transmit(uint32_t usart_periph, uint16_t data);
<b>Function descriptions</b>	USART transmit data function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>data</b>	data of transmission (0x0000-0x01FF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* USART0 transmit data */
usart_data_transmit(USART0, 0xAA);
```

### usart\_data\_receive

The description of usart\_data\_receive is shown as below:

**Table 3-1321. Function usart\_data\_receive**

<b>Function name</b>	usart_data_receive
<b>Function prototype</b>	uint16_t usart_data_receive(uint32_t usart_periph);
<b>Function descriptions</b>	USART receive data function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint16_t</b>	data of received (0x0000-0x01FF)

Example:

```
/* USART0 receive data */
uint16_t temp;
```

```
temp = usart_data_receive(USART0);
```

### usart\_command\_enable

The description of usart\_command\_enable is shown as below:

**Table 3-1322. Function usart\_command\_enable**

<b>Function name</b>	usart_command_enable
<b>Function prototype</b>	void usart_command_enable(uint32_t usart_periph, uint32_t cmdtype);
<b>Function descriptions</b>	enable USART command
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>cmdtype</b>	command type
USART_CMD_SBKCM D	send break command
USART_CMD_MMCM	mute mode command
USART_CMD_RXFCM D	receive data flush command
USART_CMD_TXFCM D	transmit data flush request
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 command */
```

```
usart_command_enable(USART0, USART_CMD_SBKCM);
```

### usart\_address\_0\_match\_mode\_enable

The description of usart\_address\_0\_match\_mode\_enable is shown as below:

**Table 3-1323. Function usart\_address\_0\_match\_mode\_enable**

<b>Function name</b>	usart_address_0_match_mode_enable
<b>Function prototype</b>	void usart_address_0_match_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable address 0 match mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable address 0 match mode */
```

```
usart_address_0_match_mode_enable (USART0);
```

### usart\_address\_0\_match\_mode\_disable

The description of usart\_address\_0\_match\_mode\_disable is shown as below:

**Table 3-1324. Function usart\_address\_0\_match\_mode\_disable**

<b>Function name</b>	usart_address_0_match_mode_disable
<b>Function prototype</b>	void usart_address_0_match_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable address 0 match mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable address 0 match mode */
```

```
usart_address_0_match_mode_disable(USART0);
```

### usart\_address\_1\_match\_mode\_enable

The description of usart\_address\_1\_match\_mode\_enable is shown as below:

**Table 3-1325. Function usart\_address\_1\_match\_mode\_enable**

<b>Function name</b>	usart_address_1_match_mode_enable
<b>Function prototype</b>	void usart_address_1_match_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable address 1 match mode
<b>Precondition</b>	-

The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable address 1 match mode */
```

```
usart_address_1_match_mode_enable(USART0);
```

### usart\_address\_1\_match\_mode\_disable

The description of usart\_address\_1\_match\_mode\_disable is shown as below:

**Table 3-1326. Function usart\_address\_1\_match\_mode\_disable**

Function name	usart_address_1_match_mode_disable
Function prototype	void usart_address_1_match_mode_disable(uint32_t usart_periph);
Function descriptions	disable address 1 match mode
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable address 1 match mode */
```

```
usart_address_1_match_mode_disable(USART0);
```

### usart\_address\_0\_config

The description of usart\_address\_0\_config is shown as below:

**Table 3-1327. Function usart\_address\_0\_config**

Function name	usart_address_0_config
Function prototype	void usart_address_0_config(uint32_t usart_periph, uint8_t addr);

<b>Function descriptions</b>	configure address 0 of the USART
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>addr</b>	address of USART (0x00-0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure address 0 of the USART0 */
usart_address_0_config(USART0, 0x00);
```

### usart\_address\_1\_config

The description of usart\_address\_1\_config is shown as below:

**Table 3-1328. Function usart\_address\_1\_config**

<b>Function name</b>	usart_address_1_config
<b>Function prototype</b>	void usart_address_1_config(uint32_t usart_periph, uint8_t addr);
<b>Function descriptions</b>	configure address 1 of the USART
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>addr</b>	address of USART (0x00-0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure address 1 of the USART0 */
usart_address_1_config(USART0, 0x00);
```

## usart\_address\_0\_detection\_mode\_config

The description of usart\_address\_0\_detection\_mode\_config is shown as below:

**Table 3-1329. Function usart\_address\_0\_detection\_mode\_config**

<b>Function name</b>	usart_address_0_detection_mode_config
<b>Function prototype</b>	void usart_address_0_detection_mode_config(uint32_t usart_periph, uint32_t addmod);
<b>Function descriptions</b>	configure address 0 detection mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>addmod</b>	address detection mode
<i>USART_ADDM_4BIT</i>	4 bits
<i>USART_ADDM_FULLBIT</i>	full bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/*configure address 0 detection mode */
```

```
usart_address_0_detection_mode_config(USART0, USART_ADDM_4BIT);
```

## usart\_address\_1\_detection\_mode\_config

The description of usart\_address\_1\_detection\_mode\_config is shown as below:

**Table 3-1330. Function usart\_address\_1\_detection\_mode\_config**

<b>Function name</b>	usart_address_1_detection_mode_config
<b>Function prototype</b>	void usart_address_1_detection_mode_config(uint32_t usart_periph, uint32_t addmod);
<b>Function descriptions</b>	configure address 1 detection mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7

Input parameter{in}	
<b>addmod</b>	address detection mode
<i>USART_ADDM_4BIT</i>	4 bits
<i>USART_ADDM_FULLBIT</i>	full bits
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/*configure address 1 detection mode */
```

```
usart_address_1_detection_mode_config(USART0, USART_ADDM_4BIT);
```

### usart\_mute\_mode\_enable

The description of usart\_mute\_mode\_enable is shown as below:

**Table 3-1331. Function usart\_mute\_mode\_enable**

<b>Function name</b>	usart_mute_mode_enable
<b>Function prototype</b>	void usart_mute_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable mute mode
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 receiver in mute mode */
```

```
usart_mute_mode_enable(USART0);
```

### usart\_mute\_mode\_disable

The description of usart\_mute\_mode\_disable is shown as below:

**Table 3-1332. Function usart\_mute\_mode\_disable**

<b>Function name</b>	usart_mute_mode_disable
<b>Function prototype</b>	void usart_mute_mode_disable(uint32_t usart_periph);

<b>Function descriptions</b>	disable mute mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 mute mode */
```

```
usart_mute_mode_disable(USART0);
```

### usart\_mute\_mode\_wakeup\_config

The description of usart\_mute\_mode\_wakeup\_config is shown as below:

**Table 3-1333. Function usart\_mute\_mode\_wakeup\_config**

<b>Function name</b>	usart_mute_mode_wakeup_config
<b>Function prototype</b>	void usart_mute_mode_wakeup_config(uint32_t usart_periph, uint32_t wmethod);
<b>Function descriptions</b>	configure wakeup method in mute mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>wmethod</b>	two methods be used to enter or exit the mute mode
<i>USART_WM_IDLE</i>	idle line
<i>USART_WM_ADDR</i>	address mask
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 wakeup method in mute mode */
```

```
usart_mute_mode_wakeup_config(USART0, USART_WM_IDLE);
```



## usart\_lin\_mode\_enable

The description of usart\_lin\_mode\_enable is shown as below:

**Table 3-1334. Function usart\_lin\_mode\_enable**

<b>Function name</b>	usart_lin_mode_enable
<b>Function prototype</b>	void usart_lin_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable LIN mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 LIN mode */
usart_lin_mode_enable(USART0);
```

## usart\_lin\_mode\_disable

The description of usart\_lin\_mode\_disable is shown as below:

**Table 3-1335. Function usart\_lin\_mode\_disable**

<b>Function name</b>	usart_lin_mode_disable
<b>Function prototype</b>	void usart_lin_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable LIN mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 LIN mode */
usart_lin_mode_disable(USART0);
```

## usart\_lin\_break\_dection\_length\_config

The description of usart\_lin\_break\_dection\_length\_config is shown as below:

**Table 3-1336. Function usart\_lin\_break\_dection\_length\_config**

<b>Function name</b>	usart_lin_break_dection_length_config
<b>Function prototype</b>	void usart_lin_break_dection_length_config(uint32_t usart_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>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2, 5
<b>Input parameter{in}</b>	
<b>lblen</b>	two methods be used to enter or exit the mute mode
USART_LBLEN_10B	10 bits
USART_LBLEN_11B	11 bits
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure LIN break frame length */
```

```
usart_lin_break_dection_length_config(USART0, USART_LBLEN_10B);
```

## usart\_halfduplex\_enable

The description of usart\_halfduplex\_enable is shown as below:

**Table 3-1337. Function usart\_halfduplex\_enable**

<b>Function name</b>	usart_halfduplex_enable
<b>Function prototype</b>	void usart_halfduplex_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable half-duplex mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* enable USART0 half duplex mode */
```

```
usart_halfduplex_enable(USART0);
```

### usart\_halfduplex\_disable

The description of usart\_halfduplex\_disable is shown as below:

**Table 3-1338. Function usart\_halfduplex\_disable**

<b>Function name</b>	usart_halfduplex_disable
<b>Function prototype</b>	void usart_halfduplex_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable half-duplex mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 half duplex mode */
```

```
usart_halfduplex_disable(USART0);
```

### usart\_clock\_enable

The description of usart\_clock\_enable is shown as below:

**Table 3-1339. Function usart\_clock\_enable**

<b>Function name</b>	usart_clock_enable
<b>Function prototype</b>	void usart_clock_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable clock
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* enable clock */
```

```
usart_clock_enable(USART0);
```

### usart\_clock\_disable

The description of usart\_clock\_disable is shown as below:

**Table 3-1340. Function usart\_clock\_disable**

Function name	usart_clock_disable
Function prototype	void usart_clock_disable(uint32_t usart_periph);
Function descriptions	disable clock
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable clock */
```

```
usart_clock_disable(USART0);
```

### usart\_synchronous\_clock\_config

The description of usart\_synchronous\_clock\_config is shown as below:

**Table 3-1341. Function usart\_synchronous\_clock\_config**

Function name	usart_synchronous_clock_config
Function prototype	void usart_synchronous_clock_config(uint32_t usart_periph, uint32_t clen, uint32_t cph, uint32_t cpl);
Function descriptions	configure USART synchronous mode parameters
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	usart peripheral
USARTx	x = 0, 1, 2, 5
Input parameter{in}	

<b>clen</b>	last bit clock pulse
<i>USART_CLEN_NONE</i>	clock pulse of the last data bit (MSB) is not output to the CK pin
<i>USART_CLEN_EN</i>	clock pulse of the last data bit (MSB) is output to the CK pin
<b>Input parameter{in}</b>	
<b>cph</b>	clock phase
<i>USART_CPH_1CK</i>	first clock transition is the first data capture edge
<i>USART_CPH_2CK</i>	second clock transition is the first data capture edge
<b>Input parameter{in}</b>	
<b>cpl</b>	clock polarity
<i>USART_CPL_LOW</i>	steady low value on CK pin
<i>USART_CPL_HIGH</i>	steady high value on CK pin
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 synchronous mode parameters */
```

```
usart_synchronous_clock_config(USART0, USART_CLEN_EN, USART_CPH_2CK,
USART_CPL_HIGH);
```

### usart\_guard\_time\_config

The description of usart\_guard\_time\_config is shown as below:

**Table 3-1342. Function usart\_guard\_time\_config**

<b>Function name</b>	usart_guard_time_config
<b>Function prototype</b>	void usart_guard_time_config(uint32_t usart_periph,uint32_t guat);
<b>Function descriptions</b>	configure guard time value in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Input parameter{in}</b>	
<b>guat</b>	guard time value (0x00000000-0x000000FF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 guard time value in smartcard mode */
```

```
usart_guard_time_config(USART0, 0x00000055);
```

### usart\_smartcard\_mode\_enable

The description of usart\_smartcard\_mode\_enable is shown as below:

**Table 3-1343. Function usart\_smartcard\_mode\_enable**

<b>Function name</b>	usart_smartcard_mode_enable
<b>Function prototype</b>	void usart_smartcard_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 smartcard mode */
usart_smartcard_mode_enable(USART0);
```

### usart\_smartcard\_mode\_disable

The description of usart\_smartcard\_mode\_disable is shown as below:

**Table 3-1344. Function usart\_smartcard\_mode\_disable**

<b>Function name</b>	usart_smartcard_mode_disable
<b>Function prototype</b>	void usart_smartcard_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 smartcard mode */
```

```
usart_smartcard_mode_disable(USART0);
```

### usart\_smartcard\_mode\_nack\_enable

The description of usart\_smartcard\_mode\_nack\_enable is shown as below:

**Table 3-1345. Function usart\_smartcard\_mode\_nack\_enable**

<b>Function name</b>	usart_smartcard_mode_nack_enable
<b>Function prototype</b>	void usart_smartcard_mode_nack_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable NACK in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 NACK in smartcard mode */
usart_smartcard_mode_nack_enable(USART0);
```

### usart\_smartcard\_mode\_nack\_disable

The description of usart\_smartcard\_mode\_nack\_disable is shown as below:

**Table 3-1346. Function usart\_smartcard\_mode\_nack\_disable**

<b>Function name</b>	usart_smartcard_mode_nack_disable
<b>Function prototype</b>	void usart_smartcard_mode_nack_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable NACK in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 NACK in smartcard mode */
```

```
usart_smartcard_mode_nack_disable(USART0);
```

### usart\_smartcard\_mode\_early\_nack\_enable

The description of usart\_smartcard\_mode\_early\_nack\_enable is shown as below:

**Table 3-1347. Function usart\_smartcard\_mode\_early\_nack\_enable**

<b>Function name</b>	usart_smartcard_mode_early_nack_enable
<b>Function prototype</b>	void usart_smartcard_mode_early_nack_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable early NACK in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 early NACK in smartcard mode */
usart_smartcard_mode_early_nack_enable(USART0);
```

### usart\_smartcard\_mode\_early\_nack\_disable

The description of usart\_smartcard\_mode\_early\_nack\_disable is shown as below:

**Table 3-1348. Function usart\_smartcard\_mode\_early\_nack\_disable**

<b>Function name</b>	usart_smartcard_mode_early_nack_disable
<b>Function prototype</b>	void usart_smartcard_mode_early_nack_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable early NACK in smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 early NACK in smartcard mode */
```



```
usart_smartcard_mode_early_nack_disable(USART0);
```

### usart\_smartcard\_autoretry\_config

The description of usart\_smartcard\_autoretry\_config is shown as below:

**Table 3-1349. Function usart\_smartcard\_autoretry\_config**

<b>Function name</b>	usart_smartcard_autoretry_config
<b>Function prototype</b>	void usart_smartcard_autoretry_config(uint32_t usart_periph, uint32_t scrtnum);
<b>Function descriptions</b>	configure smartcard auto-retry number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Input parameter{in}</b>	
<b>scrtnum</b>	smartcard auto-retry number (0x00000000-0x00000007)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure smartcard auto-retry number */
```

```
usart_smartcard_autoretry_config(USART0, 0x00000007);
```

### usart\_block\_length\_config

The description of usart\_block\_length\_config is shown as below:

**Table 3-1350. Function usart\_block\_length\_config**

<b>Function name</b>	usart_block_length_config
<b>Function prototype</b>	void usart_block_length_config(uint32_t usart_periph, uint32_t bl);
<b>Function descriptions</b>	configure block length
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Input parameter{in}</b>	
<b>bl</b>	block length(0x00000000-0x000000FF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* configure block length in smartcard T=1 reception */
```

```
usart_block_length_config(USART0, 0x000000FF);
```

### usart\_irda\_mode\_enable

The description of usart\_irda\_mode\_enable is shown as below:

**Table 3-1351. Function usart\_irda\_mode\_enable**

<b>Function name</b>	usart_irda_mode_enable
<b>Function prototype</b>	void usart_irda_mode_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable IrDA mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 IrDA mode */
```

```
usart_irda_mode_enable(USART0);
```

### usart\_irda\_mode\_disable

The description of usart\_irda\_mode\_disable is shown as below:

**Table 3-1352. Function usart\_irda\_mode\_disable**

<b>Function name</b>	usart_irda_mode_disable
<b>Function prototype</b>	void usart_irda_mode_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable IrDA mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	

-	-
---	---

Example:

```
/* disable USART0 IrDA mode */
usart_irda_mode_disable(USART0);
```

### usart\_prescaler\_config

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

**Table 3-1353. Function usart\_prescaler\_config**

<b>Function name</b>	usart_prescaler_config
<b>Function prototype</b>	void usart_prescaler_config(uint32_t usart_periph, uint8_t psc);
<b>Function descriptions</b>	configure the peripheral clock prescaler in USART IrDA low-power or smartcard mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2, 5
<b>Input parameter{in}</b>	
<b>psc</b>	clock prescaler (0x00-0xFF)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure the USART0 peripheral clock prescaler in USART IrDA low-power mode */
usart_prescaler_config(USART0, 0x00);
```

### usart\_irda\_lowpower\_config

The description of usart\_irda\_lowpower\_config is shown as below:

**Table 3-1354. Function usart\_irda\_lowpower\_config**

<b>Function name</b>	usart_irda_lowpower_config
<b>Function prototype</b>	void usart_irda_lowpower_config(uint32_t usart_periph, uint32_t irlp);
<b>Function descriptions</b>	configure IrDA low-power
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2, 5

Input parameter{in}	
<b>irlp</b>	IrDA low-power or normal
<i>USART_IRLP_LOW</i>	low-power
<i>USART_IRLP_NORMAL</i>	normal
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 IrDA low-power */
```

```
usart_irda_lowpower_config(USART0, USART_IRLP_LOW);
```

### usart\_hardware\_flow\_rts\_config

The description of usart\_hardware\_flow\_rts\_config is shown as below:

**Table 3-1355. Function usart\_hardware\_flow\_rts\_config**

<b>Function name</b>	usart_hardware_flow_rts_config
<b>Function prototype</b>	void usart_hardware_flow_rts_config(uint32_t usart_periph, uint32_t rtsconfig);
<b>Function descriptions</b>	configure hardware flow control RTS
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Input parameter{in}	
<b>rtsconfig</b>	enable or disable RTS
<i>USART_RTS_ENABLE</i>	enable RTS
<i>USART_RTS_DISABLE</i>	disable RTS
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure USART0 hardware flow control RTS */
```

```
usart_hardware_flow_rts_config(USART0, USART_RTS_ENABLE);
```

## usart\_hardware\_flow\_cts\_config

The description of usart\_hardware\_flow\_cts\_config is shown as below:

**Table 3-1356. Function usart\_hardware\_flow\_cts\_config**

<b>Function name</b>	usart_hardware_flow_cts_config
<b>Function prototype</b>	void usart_hardware_flow_cts_config(uint32_t usart_periph, uint32_t ctsconfig);
<b>Function descriptions</b>	configure hardware flow control CTS
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>ctsconfig</b>	enable or disable CTS
USART_CTS_ENABLE	enable CTS
USART_CTS_DISABLE	disable CTS
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 hardware flow control CTS */
```

```
usart_hardware_flow_cts_config(USART0, USART_CTS_ENABLE);
```

## usart\_hardware\_flow\_coherence\_config

The description of usart\_hardware\_flow\_coherence\_config is shown as below:

**Table 3-1357. Function usart\_hardware\_flow\_coherence\_config**

<b>Function name</b>	usart_hardware_flow_coherence_config
<b>Function prototype</b>	void usart_hardware_flow_coherence_config(uint32_t usart_periph, uint32_t hcm);
<b>Function descriptions</b>	configure hardware flow control coherence mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7

Input parameter{in}	
<b>hcm</b>	hardware flow control coherence mode
<i>USART_HCM_NONE</i>	nRTS signal equals to the rxne status register
<i>USART_HCM_EN</i>	nRTS signal is set when the last data bit has been sampled
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure hardware flow control coherence mode */
usart_hardware_flow_coherence_config(USART0, USART_HCM_NONE);
```

### usart\_rs485\_driver\_enable

The description of usart\_rs485\_driver\_enable is shown as below:

**Table 3-1358. Function usart\_rs485\_driver\_enable**

<b>Function name</b>	usart_rs485_driver_enable
<b>Function prototype</b>	void usart_rs485_driver_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable USART RS485 driver
<b>Precondition</b>	-
<b>The called functions</b>	-
Input parameter{in}	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable USART0 RS485 driver */
usart_rs485_driver_enable(USART0);
```

### usart\_rs485\_driver\_disable

The description of usart\_rs485\_driver\_disable is shown as below:

**Table 3-1359. Function usart\_rs485\_driver\_disable**

<b>Function name</b>	usart_rs485_driver_disable
<b>Function prototype</b>	void usart_rs485_driver_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable USART RS485 driver

<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 RS485 driver */
usart_rs485_driver_disable(USART0);
```

### usart\_driver\_assertime\_config

The description of usart\_driver\_assertime\_config is shown as below:

**Table 3-1360. Function usart\_driver\_assertime\_config**

<b>Function name</b>	usart_driver_assertime_config
<b>Function prototype</b>	void usart_driver_assertime_config(uint32_t usart_periph, uint32_t deatime);
<b>Function descriptions</b>	configure driver enable assertion time
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>deatime</b>	driver enable assertion time (0x00000000-0x0000001F)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set USART0 driver assertime */
usart_driver_assertime_config(USART0, 0x0000001F);
```

### usart\_driver\_deassertime\_config

The description of usart\_driver\_deassertime\_config is shown as below:

Table 3-1361. Function `usart_driver_deasserttime_config`

Function name	<code>usart_driver_deasserttime_config</code>
Function prototype	<code>void usart_driver_deasserttime_config(uint32_t usart_periph, uint32_t dedtime);</code>
Function descriptions	configure driver enable de-assertion time
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	usart peripheral
<code>USARTx</code>	x = 0, 1, 2, 5
<code>UARTx</code>	x = 3, 4, 6, 7
Input parameter{in}	
<code>dedtime</code>	driver enable de-assertion time (0x00000000-0x0000001F)
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* set USART0 driver deasserttime */
```

```
usart_driver_deasserttime_config(USART0, 0x0000001F);
```

### `usart_depolarity_config`

The description of `usart_depolarity_config` is shown as below:

Table 3-1362. Function `usart_depolarity_config`

Function name	<code>usart_depolarity_config</code>
Function prototype	<code>void usart_depolarity_config(uint32_t usart_periph, uint32_t dep);</code>
Function descriptions	configure driver enable polarity mode
Precondition	-
The called functions	-
Input parameter{in}	
<code>usart_periph</code>	usart peripheral
<code>USARTx</code>	x = 0, 1, 2, 5
<code>UARTx</code>	x = 3, 4, 6, 7
Input parameter{in}	
<code>dep</code>	DE signal
<code>USART_DEP_HIGH</code>	DE signal is active high
<code>USART_DEP_LOW</code>	DE signal is active low
Output parameter{out}	
-	-
Return value	
-	-



Example:

```
/* configure driver enable polarity mode */
usart_depolarity_config(USART0, USART_DEP_HIGH);
```

### usart\_dma\_receive\_config

The description of usart\_dma\_receive\_config is shown as below:

**Table 3-1363. Function usart\_dma\_receive\_config**

<b>Function name</b>	usart_dma_receive_config
<b>Function prototype</b>	void usart_dma_receive_config(uint32_t usart_periph, uint32_t dmacmd);
<b>Function descriptions</b>	configure USART DMA for reception
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>dmacmd</b>	USART DMA mode
USART_RECEIVE_DMA_ENABLE	USART enable DMA for reception
USART_RECEIVE_DMA_DISABLE	USART disable DMA for reception
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* USART0 DMA enable for reception */
usart_dma_receive_config (USART0, USART_RECEIVE_DMA_ENABLE);
```

### usart\_dma\_transmit\_config

The description of usart\_dma\_transmit\_config is shown as below:

**Table 3-1364. Function usart\_dma\_transmit\_config**

<b>Function name</b>	usart_dma_transmit_config
<b>Function prototype</b>	void usart_dma_transmit_config(uint32_t usart_periph, uint32_t dmacmd);
<b>Function descriptions</b>	configure USART DMA transmission
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	

<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>dmacmd</b>	USART DMA mode
<i>USART_TRANSMIT_DMA_ENABLE</i>	USART enable DMA for transmission
<i>USART_TRANSMIT_DMA_DISABLE</i>	USART disable DMA for transmission
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* USART0 DMA disable for transmission */
```

```
usart_dma_transmit_config (USART0, USART_TRANSMIT_DMA_DISABLE);
```

### usart\_reception\_error\_dma\_disable

The description of usart\_reception\_error\_dma\_disable is shown as below:

**Table 3-1365. Function usart\_reception\_error\_dma\_disable**

<b>Function name</b>	usart_reception_error_dma_disable
<b>Function prototype</b>	void usart_reception_error_dma_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable DMA on reception error
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable DMA on reception error */
```

```
usart_reception_error_dma_disable(USART0);
```

### usart\_reception\_error\_dma\_enable

The description of usart\_reception\_error\_dma\_enable is shown as below:

**Table 3-1366. Function usart\_reception\_error\_dma\_enable**

<b>Function name</b>	usart_reception_error_dma_enable
<b>Function prototype</b>	void usart_reception_error_dma_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable DMA on reception error
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable DMA on reception error */
usart_reception_error_dma_enable(USART0);
```

### usart\_wakeup\_enable

The description of usart\_wakeup\_enable is shown as below:

**Table 3-1367. Function usart\_wakeup\_enable**

<b>Function name</b>	usart_wakeup_enable
<b>Function prototype</b>	void usart_wakeup_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable USART to wakeup the mcu from deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 wake up */
usart_wakeup_enable(USART0);
```

### usart\_wakeup\_disable

The description of usart\_wakeup\_disable is shown as below:

**Table 3-1368. Function usart\_wakeup\_disable**

<b>Function name</b>	usart_wakeup_disable
<b>Function prototype</b>	void usart_wakeup_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable USART to wakeup the mcu from deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable USART0 wake up */
usart_wakeup_disable(USART0);
```

### usart\_wakeup\_mode\_config

The description of usart\_wakeup\_mode\_config is shown as below:

**Table 3-1369. Function usart\_wakeup\_mode\_config**

<b>Function name</b>	usart_wakeup_mode_config
<b>Function prototype</b>	void usart_wakeup_mode_config(uint32_t usart_periph, uint32_t wum);
<b>Function descriptions</b>	configure the USART wakeup mode from deep-sleep mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	usart peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<b>Input parameter{in}</b>	
<b>wum</b>	wakeup mode
<i>USART_WUM_ADDR</i>	WUF active on address match
<i>USART_WUM_START B</i>	WUF active on start bit
<i>USART_WUM_RBNE</i>	WUF active on RBNE
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure USART0 wake up mode */
```

```
usart_wakeup_mode_config(USART0, USART_WUM_ADDR);
```

### usart\_fifo\_enable

The description of usart\_fifo\_enable is shown as below:

**Table 3-1370. Function usart\_fifo\_enable**

<b>Function name</b>	usart_fifo_enable
<b>Function prototype</b>	void usart_fifo_enable(uint32_t usart_periph);
<b>Function descriptions</b>	enable FIFO
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable FIFO */
```

```
usart_fifo_enable (USART0);
```

### usart\_fifo\_disable

The description of usart\_fifo\_disable is shown as below:

**Table 3-1371. Function usart\_fifo\_disable**

<b>Function name</b>	usart_fifo_disable
<b>Function prototype</b>	void usart_fifo_disable(uint32_t usart_periph);
<b>Function descriptions</b>	disable FIFO
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable FIFO */
```

```
usart_fifo_disable(USART0);
```

### usart\_transmit\_fifo\_threshold\_config

The description of usart\_transmit\_fifo\_threshold\_config is shown as below:

**Table 3-1372. Function usart\_transmit\_fifo\_threshold\_config**

Function name	usart_transmit_fifo_threshold_config
Function prototype	void usart_transmit_fifo_threshold_config(uint32_t usart_periph, uint32_t txthreshold);
Function descriptions	configure transmit FIFO threshold
Precondition	-
The called functions	-
Input parameter{in}	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
Input parameter{in}	
<b>txthreshold</b>	transmit FIFO threshold
USART_TFTCFG_THRESHOLD_1_8	transmit FIFO reaches 1/8 of its depth
USART_TFTCFG_THRESHOLD_1_4	transmit FIFO reaches 1/4 of its depth
USART_TFTCFG_THRESHOLD_1_2	transmit FIFO reaches 1/8 of its depth
USART_TFTCFG_THRESHOLD_3_4	transmit FIFO reaches 3/4 of its depth
USART_TFTCFG_THRESHOLD_7_8	transmit FIFO reaches 7/8 of its depth
USART_TFTCFG_THRESHOLD_EMPTY	transmit FIFO becomes empty
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure transmit FIFO threshold as empty */
```

```
usart_transmit_fifo_threshold_config (USART0, USART_TFTCFG_THRESHOLD_EMPTY);
```

## usart\_receive\_fifo\_threshold\_config

The description of usart\_receive\_fifo\_threshold\_config is shown as below:

**Table 3-1373. Function usart\_receive\_fifo\_threshold\_config**

<b>Function name</b>	usart_receive_fifo_threshold_config
<b>Function prototype</b>	void usart_receive_fifo_threshold_config(uint32_t usart_periph, uint32_t rxthreshold);
<b>Function descriptions</b>	configure receive FIFO threshold
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x = 0, 1, 2, 5
UARTx	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>rxthreshold</b>	receive FIFO threshold
USART_RFTCFG_THRESHOLD_1_8	receive FIFO reaches 1/8 of its depth
USART_RFTCFG_THRESHOLD_1_4	receive FIFO reaches 1/4 of its depth
USART_RFTCFG_THRESHOLD_1_2	receive FIFO reaches 1/2 of its depth
USART_RFTCFG_THRESHOLD_3_4	receive FIFO reaches 3/4 of its depth
USART_RFTCFG_THRESHOLD_7_8	receive FIFO reaches 7/8 of its depth
USART_RFTCFG_THRESHOLD_FULL	receive FIFO becomes full
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure receiveFIFO threshold as full */
```

```
usart_receive_fifo_threshold_config (USART0, USART_RFTCFG_THRESHOLD_FULL);
```

## usart\_receive\_fifo\_counter\_number

The description of usart\_receive\_fifo\_counter\_number is shown as below:

**Table 3-1374. Function usart\_receive\_fifo\_counter\_number**

<b>Function name</b>	usart_receive_fifo_counter_number
<b>Function prototype</b>	uint8_t usart_receive_fifo_counter_number(uint32_t usart_periph);
<b>Function descriptions</b>	read receive FIFO counter number
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>uint8_t</b>	receive FIFO counter number

Example:

```
/* read receive FIFO counter number */

uint8_t temp;

temp = usart_receive_fifo_counter_number(USART0);
```

### usart\_flag\_get

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

**Table 3-1375. Function usart\_flag\_get**

<b>Function name</b>	usart_flag_get
<b>Function prototype</b>	FlagStatus usart_flag_get(uint32_t usart_periph, usart_flag_enum flag);
<b>Function descriptions</b>	get flag in STAT/CHC/FCS register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x = 0, 1, 2, 5
<i>UARTx</i>	x = 3, 4, 6, 7
<b>Input parameter{in}</b>	
<b>flag</b>	USART flags, refer to <a href="#">Table 3-1298. Enum usart_flag_enum</a> only one among these parameters can be selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:



```
/* get flag USART0 state */
```

```
FlagStatus status;
```

```
status = usart_flag_get(USART0, USART_FLAG_TBE);
```

### usart\_flag\_clear

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

**Table 3-1376. Function usart\_flag\_clear**

<b>Function name</b>	usart_flag_clear
<b>Function prototype</b>	void usart_flag_clear(uint32_t usart_periph, usart_flag_enum flag);
<b>Function descriptions</b>	clear flag in STAT register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
USARTx	x=0,1,2
<b>Input parameter{in}</b>	
<b>flag</b>	USART flags, refer to <a href="#">Table 3-1298. Enum usart_flag_enum</a> only one among these parameters can be selected
USART_FLAG_PERR	parity error flag
USART_FLAG_FERR	frame error flag
USART_FLAG_NERR	noise detected flag
USART_FLAG_ORER R	overrun error flag
USART_FLAG_IDLE	idle line detected flag
USART_FLAG_TC	transmission complete flag
USART_FLAG_LBD	LIN break detected flag
USART_FLAG_CTSF	CTS change flag
USART_FLAG_RT	receiver timeout flag
USART_FLAG_EB	end of block flag
USART_FLAG_AM0	address 0 match flag
USART_FLAG_AM1	address 1 match flag
USART_FLAG_WU	wakeup from deep-sleep mode flag
USART_FLAG_EPERR	early parity error flag
USART_FLAG_TFE	transmit FIFO empty flag
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear USART0 flag */
```

```
usart_flag_clear(USART0, USART_FLAG_TC);
```

## usart\_interrupt\_enable

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

**Table 3-1377. Function usart\_interrupt\_enable**

<b>Function name</b>	usart_interrupt_enable
<b>Function prototype</b>	void usart_interrupt_enable(uint32_t usart_periph, usart_interrupt_enum interrupt);
<b>Function descriptions</b>	enable USART interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>interrupt</b>	interrupt type, refer to <a href="#">Table 3-1300. Enum usart_interrupt_enum</a> only one among these parameters can be selected
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable USART0 TBE interrupt */
```

```
usart_interrupt_enable(USART0, USART_INT_TBE);
```

## usart\_interrupt\_disable

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

**Table 3-1378. Function usart\_interrupt\_disable**

<b>Function name</b>	usart_interrupt_disable
<b>Function prototype</b>	void usart_interrupt_disable(uint32_t usart_periph, usart_interrupt_enum interrupt);
<b>Function descriptions</b>	disable USART interrupt
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<i>USARTx</i>	x=0,1,2
<b>Input parameter{in}</b>	
<b>interrupt</b>	interrupt type, refer to <a href="#">Table 3-1300. Enum usart_interrupt_enum</a> only one among these parameters can be selected

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* disable USART0 TBE interrupt */
```

```
usart_interrupt_disable(USART0, USART_INT_TBE);
```

### usart\_interrupt\_flag\_get

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

**Table 3-1379. Function usart\_interrupt\_flag\_get**

Function name	usart_interrupt_flag_get
Function prototype	FlagStatus usart_interrupt_flag_get(uint32_t usart_periph, usart_interrupt_flag_enum int_flag);
Function descriptions	get USART interrupt and flag status
Precondition	-
The called functions	-
Input parameter{in}	
usart_periph	USARTx peripheral
USARTx	x=0,1,2
Input parameter{in}	
int_flag	USART interrupt flag, refer to <a href="#">Table 3-1299. Enum usart_interrupt_flag_enum</a> , only one among these parameters can be selected
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the USART0 interrupt flag status */
```

```
FlagStatus status;
```

```
status = usart_interrupt_flag_get(USART0, USART_INT_FLAG_RBNE);
```

### usart\_interrupt\_flag\_clear

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

**Table 3-1380. Function usart\_interrupt\_flag\_clear**

Function name	usart_interrupt_flag_clear
Function prototype	void usart_interrupt_flag_clear(uint32_t usart_periph,

	usart_interrupt_flag_enum int_flag);
<b>Function descriptions</b>	clear USART interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>usart_periph</b>	USARTx peripheral
<b>USARTx</b>	x=0,1,2
<b>Input parameter{in}</b>	
<b>int_flag</b>	USART interrupt flag, refer to <a href="#">Table 3-1299. Enum usart_interrupt_flag_enum</a> , only one among these parameters can be selected
USART_INT_FLAG_PERRR	parity error flag
USART_INT_FLAG_ERRFR	frame error flag
USART_INT_FLAG_ERRNR	noise detected flag
USART_INT_FLAG_RBNE_ORERR	read data buffer not empty interrupt and overrun error flag
USART_INT_FLAG_ERRORERR	error interrupt and overrun error
USART_INT_FLAG_IDLEL	idle line detected flag
USART_INT_FLAG_TC	transmission complete flag
USART_INT_FLAG_LBD	LIN break detected flag
USART_INT_FLAG_CTS	CTS change flag
USART_INT_FLAG_RRT	receiver timeout flag
USART_INT_FLAG_EB	end of block flag
USART_INT_FLAG_A0M	address 0 match flag
USART_INT_FLAG_A1M	address 1 match flag
USART_INT_FLAG_WU	wakeup from deep-sleep mode flag
USART_INT_RFT	receive FIFO threshold interrupt
USART_INT_FLAG_RFF	receive FIFO full interrupt and flag
USART_INT_FLAG_TFE	transmit FIFO empty interrupt and flag
<b>Output parameter{out}</b>	
-	-

Return value	
-	-

Example:

```
/* clear the USART0 interrupt flag */
```

```
usart_interrupt_flag_clear(USART0, USART_INT_FLAG_TC);
```

## 3.37. VREF

The precision internal reference is used to provide reference voltage for ADC / DAC, or used by off-chip circuit connecting to V<sub>REFP</sub> pin. The VREF registers are listed in chapter [3.37.1](#), the VREF firmware functions are introduced in chapter [3.37.2](#).

### 3.37.1. Descriptions of Peripheral registers

VREF registers are listed in the table shown as below:

**Table 3-1381. VREF Registers**

Registers	Descriptions
VREF_CS	VREF control and status register
VREF_CALIB	VREF calibration register

### 3.37.2. Descriptions of Peripheral functions

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

**Table 3-1382. VREF firmware function**

Function name	Function description
vref_deinit	Deinitialize the VREF
vref_enable	Enable VREF
vref_disable	Disable VREF
vref_high_impedance_mode_enable	Enable VREF high impedance mode
vref_high_impedance_mode_disable	Disable VREF high impedance mode
vref_status_get	Get the status of VREF
vref_voltage_select	Select the VREF voltage reference
vref_calib_value_set	Set the calibration value of VREF
vref_calib_value_get	Get the calibration value of VREF

#### vref\_deinit

The description of vref\_deinit is shown as below:

**Table 3-1383. Function vref\_deinit**

Function name	vref_deinit
---------------	-------------

<b>Function prototype</b>	void vref_deinit(void);
<b>Function descriptions</b>	Deinitialize the VREF
<b>Precondition</b>	-
<b>The called functions</b>	rcu_periph_reset_enable / rcu_periph_reset_disable
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* deinitialize the VREF */
vref_deinit();
```

### vref\_enable

The description of vref\_enable is shown as below:

**Table 3-1384. Function vref\_enable**

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

Example:

```
/* enable VREF */
vref_enable();
```

### vref\_disable

The description of vref\_disable is shown as below:

**Table 3-1385. Function vref\_disable**

<b>Function name</b>	vref_disable
<b>Function prototype</b>	void vref_disable(void);
<b>Function descriptions</b>	Disable VREF

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

Example:

```
/* disable VREF */
```

```
vref_disable();
```

### vref\_high\_impedance\_mode\_enable

The description of vref\_high\_impedance\_mode\_enable is shown as below:

**Table 3-1386. Function vref\_high\_impedance\_mode\_enable**

Function name	vref_high_impedance_mode_enable
Function prototype	void vref_high_impedance_mode_enable(void);
Function descriptions	Enable VREF high impedance mode
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable VREF high impedance mode */
```

```
vref_high_impedance_mode_enable();
```

### vref\_high\_impedance\_mode\_disable

The description of vref\_high\_impedance\_mode\_disable is shown as below:

**Table 3-1387. Function vref\_high\_impedance\_mode\_disable**

Function name	vref_high_impedance_mode_disable
Function prototype	void vref_high_impedance_mode_disable(void);
Function descriptions	Disable VREF high impedance mode
Precondition	-
The called functions	-

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

Example:

```
/* disable VREF high impedance mode */
```

```
vref_high_impedance_mode_disable();
```

### vref\_status\_get

The description of vref\_status\_get is shown as below:

**Table 3-1388. Function vref\_status\_get**

Function name	vref_status_get
Function prototype	FlagStatus vref_status_get(void);
Function descriptions	Get the status of VREF
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
FlagStatus	SET or RESET

Example:

```
/* get the status of VREF */
```

```
FlagStatus status;
```

```
status = vref_status_get();
```

### vref\_voltage\_select

The description of vref\_voltage\_select is shown as below:

**Table 3-1389. Function vref\_voltage\_select**

Function name	vref_voltage_select
Function prototype	void vref_voltage_select(uint32_t vref_voltage);
Function descriptions	Select the VREF voltage reference
Precondition	-
The called functions	-
Input parameter{in}	



<b>vref_voltage</b>	VREF voltage reference select
VREF_VOLTAGE_SEL_2_5V	VREF voltage reference select 2.5 V
VREF_VOLTAGE_SEL_2_048V	VREF voltage reference select 2.048 V
VREF_VOLTAGE_SEL_1_8V	VREF voltage reference select 1.8 V
VREF_VOLTAGE_SEL_1_5V	VREF voltage reference select 1.5 V
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* select 2.5V as the VREF voltage reference */
vref_voltage_select(VREF_VOLTAGE_SEL_2_5V);
```

### vref\_calib\_value\_set

The description of vref\_calib\_value\_set is shown as below:

**Table 3-1390. Function vref\_calib\_value\_set**

<b>Function name</b>	vref_calib_value_set
<b>Function prototype</b>	void vref_calib_value_set(uint8_t value);
<b>Function descriptions</b>	Set the calibration value of VREF
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>value</b>	Calibration value (0x00 - 0x3F)
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* set the calibration value of VREF */
vref_calib_value_set(0x0A);
```

### vref\_calib\_value\_get

The description of vref\_calib\_value\_get is shown as below:

**Table 3-1391. Function vref\_calib\_value\_get**

<b>Function name</b>	vref_calib_value_get
<b>Function prototype</b>	uint8_t vref_calib_value_get(void);
<b>Function descriptions</b>	Get the calibration value of VREF
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint8_t	Calibration value (0x00 - 0x3F)

Example:

```
/* get the calibration value of VREF */

uint8_t cal_val;

cal_val = vref_calib_value_get();
```

## 3.38. WWDGT

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. The WWDGT registers are listed in chapter [3.38.1](#), the WWDGT firmware functions are introduced in chapter [3.38.2](#).

### 3.38.1. Descriptions of Peripheral registers

WWDGT registers are listed in the table shown as below:

**Table 3-1392. WWDGT Registers**

Registers	Descriptions
WWDGT_CTL	WWDGT control register
WWDGT_CFG	WWDGT configuration register
WWDGT_STAT	WWDGT status register

### 3.38.2. Descriptions of Peripheral functions

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

**Table 3-1393. WWDGT firmware function**

Function name	Function description
wwdgt_deinit	reset the window watchdog timer configuration
wwdgt_enable	start the window watchdog timer counter
wwdgt_counter_update	configure the window watchdog timer counter value

Function name	Function description
wwdgt_config	configure counter value, window value, and prescaler divider value
wwdgt_interrupt_enable	enable early wakeup interrupt of WWDGT
wwdgt_flag_get	check early wakeup interrupt state of WWDGT
wwdgt_flag_clear	clear early wakeup interrupt state of WWDGT

### wwdgt\_deinit

The description of wwdgt\_deinit is shown as below:

**Table 3-1394. Function wwdgt\_deinit**

Function name	wwdgt_deinit
Function prototype	void wwdgt_deinit(void);
Function descriptions	reset the window watchdog timer configuration
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* reset the window watchdog timer configuration */
```

```
wwdgt_deinit();
```

### wwdgt\_enable

The description of wwdgt\_enable is shown as below:

**Table 3-1395. Function wwdgt\_enable**

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

Example:

```
/* start the WWDGT counter */
```

```
wwdgt_enable();
```

### wwdgt\_counter\_update

The description of wwdgt\_counter\_update is shown as below:

**Table 3-1396. Function wwdgt\_counter\_update**

<b>Function name</b>	wwdgt_counter_update
<b>Function prototype</b>	void wwdgt_counter_update(uint16_t counter_value);
<b>Function descriptions</b>	configure the window watchdog timer counter value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>counter_value</b>	counter_value: 0x0000 - 0x007F
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* update WWDGT counter to 0x7F */
```

```
wwdgt_counter_update(127);
```

### wwdgt\_config

The description of wwdgt\_config is shown as below:

**Table 3-1397. Function wwdgt\_config**

<b>Function name</b>	wwdgt_config
<b>Function prototype</b>	void wwdgt_config(uint16_t counter, uint16_t window, uint32_t prescaler);
<b>Function descriptions</b>	configure counter value, window value, and prescaler divider value
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>counter</b>	counter: 0x0000 - 0x007F
<b>Input parameter{in}</b>	
<b>window</b>	window: 0x0000 - 0x007F
<b>Input parameter{in}</b>	
<b>prescaler</b>	wwdgt prescaler value
WWDGT_CFG_PSC_D IV1	the time base of WWDGT counter = (PCLK3/4096)/1
WWDGT_CFG_PSC_D IV2	the time base of WWDGT counter = (PCLK3/4096)/2

WWDGT_CFG_PSC_D IV4	the time base of WWDGT counter = (PCLK3/4096)/4
WWDGT_CFG_PSC_D IV8	the time base of WWDGT counter = (PCLK3/4096)/8
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure WWDGT counter value to 0x7F, window value to 0x50, prescaler divider value to 8 */
```

```
wwdgt_config(127, 80, WWDGT_CFG_PSC_DIV8);
```

### wwdgt\_interrupt\_enable

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

**Table 3-1398. Function wwdgt\_interrupt\_enable**

Function name	wwdgt_interrupt_enable
Function prototype	void wwdgt_interrupt_enable(void);
Function descriptions	enable early wakeup interrupt of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable early wakeup interrupt of WWDGT */
```

```
wwdgt_interrupt_enable();
```

### wwdgt\_flag\_get

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

**Table 3-1399. Function wwdgt\_flag\_get**

Function name	wwdgt_flag_get
Function prototype	FlagStatus wwdgt_flag_get(void);
Function descriptions	check early wakeup interrupt state of WWDGT
Precondition	-

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

Example:

```
/* test if the counter value update has reached the 0x40 */
```

```
FlagStatus status;
```

```
status = wwdgt_flag_get();
```

### wwdgt\_flag\_clear

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

**Table 3-1400. Function wwdgt\_flag\_clear**

Function name	wwdgt_flag_clear
Function prototype	void wwdgt_flag_clear(void);
Function descriptions	clear early wakeup interrupt state of WWDGT
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* clear early wakeup interrupt state of WWDGT */
```

```
wwdgt_flag_clear();
```

### 3.39. ESC\_INTC

The multi-layer interrupt structure of the device is programmable and controlled by the interrupt controller (INTC). The INTC registers are listed in chapter [3.39.1](#), the INTC firmware functions are introduced in chapter [3.39.2](#).

#### 3.39.1. Descriptions of Peripheral registers

ESC\_INTC registers are listed in the table shown as below:

**Table 3-1401. ESC\_INTC Registers**

Registers	Descriptions
INTC_CTL	INTC control register
INTC_FLAG	INTC flag register
INTC_EN	INTC enable register

#### 3.39.2. Descriptions of Peripheral functions

ESC\_INTC firmware functions are listed in the table shown as below:

**Table 3-1402. ESC\_INTC firmware function**

Function name	Function description
intc_deas_config	configure INTC interrupt de-assertion interval
intc_deasc_config	configure INTC interrupt de-assertion interval clear
intc_irq_config	configure IRQ pin out enable, polarity, clock and mode
intc_interrupt_enable	enable INTC interrupt
intc_interrupt_disable	disable INTC interrupt
intc_interrupt_flag_get	get INTC interrupt flag
intc_interrupt_flag_clear	clear INTC interrupt flag
intc_deasstat_get	get interrupt de-assertion interval status
intc_irqstat_get	get internal IRQ line status

#### Enum intc\_enable\_enum

**Table 3-1403. Enum intc\_enable\_enum**

Member name	Descriptions
SW_INT_ENABLE	enable software interrupt
READY_INT_ENABLE	enable device ready interrupt
PHYB_INT_ENABLE	enable ethernet PHY B interrupt
PHYA_INT_ENABLE	enable ethernet PHY A interrupt
TIM_INT_ENABLE	enable timer interrupt
PME_INT_ENABLE	enable PME interrupt
AHB2OPB_INT_ENABLE	enable AHB2OPB bridge interrupt

ECAT_INT_ENABLE	enable etherCAT interrupt
-----------------	---------------------------

### Enum intc\_disable\_enum

**Table 3-1404. Enum intc\_disable\_enum**

Member name	Descriptions
SW_INT_DISABLE	disable software interrupt
READY_INT_DISABLE	disable device ready interrupt
PHYB_INT_DISABLE	disable ethernet PHY B interrupt
PHYA_INT_DISABLE	disable ethernet PHY A interrupt
TIM_INT_DISABLE	disable timer interrupt
PME_INT_DISABLE	disable PME interrupt
AHB2OPB_INT_DISABLE	disable AHB2OPB bridge interrupt
ECAT_INT_DISABLE	disable etherCAT interrupt

### Enum intc\_get\_flag\_enum

**Table 3-1405. Enum intc\_get\_flag\_enum**

Member name	Descriptions
SW_INT_FLAG_GET	get software interrupt flag
READY_INT_FLAG_GET	get device ready interrupt flag
PHYB_INT_FLAG_GET	get ethernet PHY B interrupt flag
PHYA_INT_FLAG_GET	get ethernet PHY A interrupt flag
TIM_INT_FLAG_GET	get timer interrupt flag
PME_INT_FLAG_GET	get PME interrupt flag
AHB2OPB_INT_FLAG_GET	get AHB2OPB bridge interrupt flag
ECAT_INT_FLAG_GET	get etherCAT interrupt flag

### Enum intc\_clear\_flag\_enum

**Table 3-1406. Enum intc\_clear\_flag\_enum**

Member name	Descriptions
SW_INT_FLAG_CLEAR	clear software interrupt flag
READY_INT_FLAG_CLEAR	clear device ready interrupt flag
TIM_INT_FLAG_CLEAR	clear timer interrupt flag
PME_INT_FLAG_CLEAR	clear PME interrupt flag



## intc\_deas\_config

The description of intc\_deas\_config is shown as below:

**Table 3-1407. Function intc\_deas\_config**

<b>Function name</b>	intc_deas_config
<b>Function prototype</b>	void intc_deas_config(uint8_t deas);
<b>Function descriptions</b>	INTC interrupt de-assertion interval configure
<b>Precondition</b>	-
<b>The called functions</b>	read_intc_register write_intc_register
<b>Input parameter{in}</b>	
<b>uint8_t</b>	configure the de-assertion interval, uints in 10us
<i>deas</i>	0x00 – 0xFF
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure INTC interrupt de-assertion interval */
```

```
intc_deas_config(0x10);
```

## intc\_deasc\_config

The description of intc\_deasc\_config is shown as below:

**Table 3-1408. Function intc\_deasc\_config**

<b>Function name</b>	intc_deasc_config
<b>Function prototype</b>	void intc_deasc_config(uint32_t deasc);
<b>Function descriptions</b>	clear INTC interrupt de-assertion interval configure
<b>Precondition</b>	-
<b>The called functions</b>	fw_debug_report_err read_intc_register write_intc_register
<b>Input parameter{in}</b>	
<b>deasc</b>	clear de-assertion interval counter
<i>INTC_DEAS</i>	no effect
<i>INTC_DEAS_CLEAR</i>	clear interrupt de-assertion interval
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear INTC interrupt de-assertion interval */
```

```
intc_deasc_config(INTC_DEAS_CLEAR);
```

## intc\_irq\_config

The description of intc\_irq\_config is shown as below:

**Table 3-1409. Function intc\_irq\_config**

<b>Function name</b>	intc_irq_config
<b>Function prototype</b>	void intc_irq_config(uint32_t irqen, uint32_t irqpol, uint32_t irqckout, uint32_t irqmode);
<b>Function descriptions</b>	IRQ pin out enable、polarity、clock and mode configure
<b>Precondition</b>	-
<b>The called functions</b>	fw_debug_report_err read_intc_register write_intc_register
<b>Input parameter{in}</b>	
<b>irqen</b>	IRQ pin output enable
<i>IRQ_DISABLE</i>	Disable IRQ pin output
<i>IRQ_ENABLE</i>	Enable IRQ pin output
<b>Input parameter{in}</b>	
<b>irqpol</b>	IRQ pin output polarity
<i>IRQ_POL_LOW</i>	IRQ pin output activation level is low
<i>IRQ_POL_HIGH</i>	IRQ pin output activation level is high
<b>Input parameter{in}</b>	
<b>irqckout</b>	IRQ clock output
<i>IRQ_NO_CKOUT</i>	No clock output
<i>IRQ_CKOUT</i>	IRQ pin output crystal oscillator clock
<b>Input parameter{in}</b>	
<b>irqmode</b>	IRQ pin output mode
<i>IRQ_OPEN_DRAIN_MODE</i>	Output open-drain mode
<i>IRQ_PUSH_PULL_MODE</i>	Output push-pull mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* config IRQ pin out enable, polarity, clock and mode */
```

```
intc_irq_config(IRQ_ENABLE, IRQ_POL_LOW, IRQ_CKOUT, IRQ_OPEN_DRAIN_MODE);
```

## intc\_interrupt\_enable

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

**Table 3-1410. Function intc\_interrupt\_enable**

<b>Function name</b>	intc_interrupt_enable
<b>Function prototype</b>	void intc_interrupt_enable(intc_enable_enum int_type);
<b>Function descriptions</b>	enable INTC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	fw_debug_report_err read_intc_register write_intc_register
<b>Input parameter{in}</b>	
<b>intc_enable_enum</b>	INTC enable, refer to <a href="#">Table 3-1403. Enum intc_enable_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable software interrupt */
intc_interrupt_enable(SW_INT_ENABLE);
```

## intc\_interrupt\_disable

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

**Table 3-1411. Function intc\_interrupt\_disable**

<b>Function name</b>	intc_interrupt_disable
<b>Function prototype</b>	void intc_interrupt_disable(intc_disable_enum int_type);
<b>Function descriptions</b>	disable INTC interrupt
<b>Precondition</b>	-
<b>The called functions</b>	fw_debug_report_err read_intc_register write_intc_register
<b>Input parameter{in}</b>	
<b>intc_disable_enum</b>	INTC disable, refer to <a href="#">Table 3-1404. Enum intc_disable_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* disable software interrupt */
```

```
intc_interrupt_disable(SW_INT_DISABLE);
```

## intc\_interrupt\_flag\_get

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

**Table 3-1412. Function intc\_interrupt\_flag\_get**

<b>Function name</b>	intc_interrupt_flag_get
<b>Function prototype</b>	FlagStatus intc_interrupt_flag_get(intc_get_flag_enum flag_type);
<b>Function descriptions</b>	get INTC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	fw_debug_report_err read_intc_register
<b>Input parameter{in}</b>	
intc_get_flag_enum	INTC flag get, refer to <a href="#">Table 3-1406. Enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
FlagStatus	RESET or SET

Example:

```
/* get software interrupt flag */

FlagStatus sw_flag;

sw_flag = intc_interrupt_flag_get(SW_INT_FLAG_GET);
```

## intc\_interrupt\_flag\_clear

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

**Table 3-1413. Function intc\_interrupt\_flag\_clear**

<b>Function name</b>	intc_interrupt_flag_clear
<b>Function prototype</b>	void intc_interrupt_flag_clear(intc_clear_flag_enum flag_type);
<b>Function descriptions</b>	clear INTC interrupt flag
<b>Precondition</b>	-
<b>The called functions</b>	fw_debug_report_err write_intc_register
<b>Input parameter{in}</b>	
intc_clear_flag_enum	INTC flag clear, refer to <a href="#">Table 3-1406. Enum intc clear flag_enum</a>
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* clear software interrupt flag */
```

```
intc_interrupt_flag_clear(SW_INT_FLAG_CLEAR);
```

### intc\_deasstat\_get

The description of intc\_deasstat\_get is shown as below:

**Table 3-1414. Function intc\_deasstat\_get**

<b>Function name</b>	intc_deasstat_get
<b>Function prototype</b>	FlagStatus intc_deasstat_get(void);
<b>Function descriptions</b>	get interrupt de-assertion interval status
<b>Precondition</b>	-
<b>The called functions</b>	read_intc_register
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	RESET or SET

Example:

```
/* get interrupt de-assertion interval status */
```

```
FlagStatus deas_flag;
```

```
deas_flag = intc_deasstat_get();
```

### intc\_irqstat\_get

The description of intc\_irqstat\_get is shown as below:

**Table 3-1415. Function intc\_irqstat\_get**

<b>Function name</b>	intc_irqstat_get
<b>Function prototype</b>	FlagStatus intc_irqstat_get(void);
<b>Function descriptions</b>	get internal IRQ line status
<b>Precondition</b>	-
<b>The called functions</b>	read_intc_register
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	RESET or SET

Example:

```
/* get internal IRQ line status */
```

```
FlagStatus irq_flag;

irq_flag = intc_irqstat_get();
```

## 3.40. ESC\_OSPI

The EtherCat project needs to support SPI/QSPI/OSPI slave module, and the MCU accesses the EtherCat chip through the SPI/QSPI/OSPI interface. The ESC\_OSPI registers are listed in chapter [3.40.1](#), the INTc firmware functions are introduced in chapter [3.40.2](#).

### 3.40.1. Descriptions of Peripheral registers

ESC\_OSPI registers are listed in the table shown as below:

**Table 3-1416. ESC\_OSPI Registers**

Registers	Descriptions
OSPI_CTL	OSPI control register
OSPI_DCFG0	OSPI device configuration register
OSPI_DCFG1	OSPI device configuration register
OSPI_STAT	OSPI status register
OSPI_STATC	OSPI status clear register
OSPI_DTLEN	OSPI data length register
OSPI_ADDR	OSPI address register
OSPI_DATA	OSPI data register
OSPI_STATMK	OSPI status mask register
OSPI_STATMATCH	OSPI status match register
OSPI_INTERVAL	OSPI interval register
OSPI_TCFG	OSPI transfer configuration register
OSPI_TIMCFG	OSPI timing configuration register
OSPI_INS	OSPI instruction register
OSPI_ALTE	OSPI alternate bytes register
OSPI_WPTCFG	OSPI wrap transfer configuration register
OSPI_WPTIMCFG	OSPI wrap timing configuration register
OSPI_WPINS	OSPI wrap instruction register
OSPI_WPALTE	OSPI wrap alternate bytes register
OSPI_WTCFG	OSPI write transfer configuration register
OSPI_WTIMCFG	OSPI write timing configuration register
OSPI_WINS	OSPI write instruction register
OSPI_WALTE	OSPI write alternate bytes register

### 3.40.2. Descriptions of Peripheral functions

ESC\_OSPI firmware functions are listed in the table shown as below:

Table 3-1417. ESC\_OSPI firmware function

Function name	Function description
ospi_hw_init	initialize OSPI/OSPIM and GPIO
ospi_enable_qspi_mode	enable qspi mode
ospi_enable_ospi_mode	enable ospi mode
ospi_reset_spi_mode	reset spi mode
ospi_write	write data
ospi_read	read data
OSPIWriteRegUsingCSR	OSPI write register using CSR
OSPIReadRegUsingCSR	OSPI read register using CSR
OSPIWritePDRamRegister	OSPI write PDRam register
OSPIReadPDRamRegister	OSPI read PDRam register
OSPIReadRegister	read register
OSPIWriteRegister	write register

## ospi\_hw\_init

The description of ospi\_hw\_init is shown as below:

Table 3-1418. Function ospi\_hw\_init

<b>Function name</b>	ospi_hw_init
<b>Function prototype</b>	void ospi_hw_init(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct);
<b>Function descriptions</b>	initialize OSPI/OSPIM and GPIO
<b>Precondition</b>	-
<b>The called functions</b>	ospi_deinit, ospim_deinit, rcu_periph_clock_enable, gpio_af_set, gpio_mode_set, gpio_output_options_set, ospim_port_sck_config, ospim_port_csn_config, ospim_port_io3_0_config, ospim_port_io7_4_config, ospi_init, ospi_enable
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	peripheral OSPIx
<b>OSPIx</b>	x=0,1
<b>Input parameter{in}</b>	
<b>ospi_struct</b>	OSPI parameter initialization struct members of the structure
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* initialize OSPI/OSPIM and GPIO */

ospi_parameter_struct ospi_struct = {0};

ospi_hw_init(OSPI1, &ospi_struct);

```

## ospi\_enable\_qspi\_mode

The description of ospi\_enable\_qspi\_mode is shown as below:

**Table 3-1419. Function ospi\_enable\_qspi\_mode**

<b>Function name</b>	ospi_enable_qspi_mode
<b>Function prototype</b>	void ospi_enable_qspi_mode(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, interface_mode mode);
<b>Function descriptions</b>	enable qspi mode
<b>Precondition</b>	-
<b>The called functions</b>	ospi_command_config
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	peripheral OSPIx
<i>OSPIx</i>	x=0,1
<b>Input parameter{in}</b>	
<b>ospi_struct</b>	OSPI parameter initialization struct members of the structure
<b>Input parameter{in}</b>	
<b>interface_mode</b>	SPI interface
<i>SPI_MODE</i>	standard SPI mode
<i>QSPI_MODE</i>	QSPI mode
<i>OSPI_MODE</i>	OSPI mode
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable qspi mode */
ospi_parameter_struct ospi_struct = {0};
ospi_enable_qspi_mode (OSPI1, &ospi_struct, SPI_MODE);
```

## ospi\_enable\_ospi\_mode

The description of ospi\_enable\_ospi\_mode is shown as below:

**Table 3-1420. Function ospi\_enable\_ospi\_mode**

<b>Function name</b>	ospi_enable_ospi_mode
<b>Function prototype</b>	void ospi_enable_ospi_mode(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, interface_mode mode);
<b>Function descriptions</b>	enable ospi mode
<b>Precondition</b>	-
<b>The called</b>	ospi_command_config



functions	
Input parameter{in}	
<b>ospi_periph</b>	peripheral OSPIx
<i>OSPIx</i>	x=0,1
Input parameter{in}	
<b>ospi_struct</b>	OSPI parameter initialization struct members of the structure
Input parameter{in}	
<b>interface_mode</b>	SPI interface
<i>SPI_MODE</i>	standard SPI mode
<i>QSPI_MODE</i>	QSPI mode
<i>OSPI_MODE</i>	OSPI mode
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ospi mode */
```

```
ospi_parameter_struct ospi_struct = {0};
```

```
ospi_enable_ospi_mode (OSPI1, &ospi_struct, SPI_MODE);
```

### ospi\_reset\_spi\_mode

The description of ospi\_reset\_spi\_mode is shown as below:

**Table 3-1421. Function ospi\_reset\_spi\_mode**

<b>Function name</b>	ospi_reset_spi_mode
<b>Function prototype</b>	void ospi_reset_spi_mode(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, interface_mode mode);
<b>Function descriptions</b>	reset spi mode
<b>Precondition</b>	-
<b>The called functions</b>	ospi_command_config
Input parameter{in}	
<b>ospi_periph</b>	peripheral OSPIx
<i>OSPIx</i>	x=0,1
Input parameter{in}	
<i>ospi_struct</i>	OSPI parameter initialization struct members of the structure
Input parameter{in}	
<b>interface_mode</b>	SPI interface
<i>SPI_MODE</i>	standard SPI mode
<i>QSPI_MODE</i>	QSPI mode
<i>OSPI_MODE</i>	OSPI mode

Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* enable ospi mode */
ospi_parameter_struct ospi_struct = {0};
ospi_reset_spi_mode(OSPI1, &ospi_struct, SPI_MODE);
```

## ospi\_write

The description of ospi\_write is shown as below:

**Table 3-1422. Function ospi\_write**

<b>Function name</b>	ospi_write
<b>Function prototype</b>	void ospi_write(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, interface_mode mode, uint8_t cmd, addr_inc_mode inc_mode, uint32_t addr, uint8_t *pdata, uint32_t data_size);
<b>Function descriptions</b>	write data
<b>Precondition</b>	-
<b>The called functions</b>	ospi_command_config, ospi_transmit
Input parameter{in}	
<b>ospi_periph</b>	peripheral OSPIx
OSPIx	x=0,1
Input parameter{in}	
<b>ospi_struct</b>	OSPI parameter initialization struct members of the structure
Input parameter{in}	
<b>interface_mode</b>	SPI interface
SPI_MODE	standard SPI mode
QSPI_MODE	QSPI mode
OSPI_MODE	OSPI mode
Input parameter{in}	
<b>cmd</b>	command
Input parameter{in}	
<b>inc_mode</b>	address mode
ADDR_NO_INC	static address
ADDR_INC	incrementing address
ADDR_DEC	decrementing address
Input parameter{in}	
<b>addr</b>	write start address
Input parameter{in}	

<i>pdata</i>	pointer to data to be written
<b>Input parameter{in}</b>	
<i>data_size</i>	size of data to write
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write data */

uint32_t temp = 0U;

ospi_parameter_struct ospi_struct = {0};

ospi_write(OSPI_INTERFACE, &ospi_struct, OSPI_MODE, 0x01U, 0x4000U, 0x1000U,
(uint8_t *)&temp, 4);
```

## ospi\_read

The description of ospi\_read is shown as below:

**Table 3-1423. Function ospi\_read**

<b>Function name</b>	ospi_read
<b>Function prototype</b>	void ospi_read(uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, interface_mode mode, uint8_t cmd, addr_inc_mode inc_mode, uint32_t addr, uint8_t *pdata, uint32_t data_size);
<b>Function descriptions</b>	read data
<b>Precondition</b>	-
<b>The called functions</b>	ospi_command_config, ospi_transmit
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	peripheral OSPIx
<i>OSPIx</i>	x=0,1
<b>Input parameter{in}</b>	
<i>ospi_struct</i>	OSPI parameter initialization struct members of the structure
<b>Input parameter{in}</b>	
<b>interface_mode</b>	SPI interface
<i>SPI_MODE</i>	standard SPI mode
<i>QSPI_MODE</i>	QSPI mode
<i>OSPI_MODE</i>	OSPI mode
<b>Input parameter{in}</b>	
<i>cmd</i>	command
<b>Input parameter{in}</b>	
<b>inc_mode</b>	address mode
<i>ADDR_NO_INC</i>	static address

<i>ADDR_INC</i>	incrementing address
<i>ADDR_DEC</i>	decrementing address
<b>Input parameter{in}</b>	
<i>addr</i>	read start address
<b>Input parameter{in}</b>	
<i>pdata</i>	pointer to data to be read
<b>Input parameter{in}</b>	
<i>data_size</i>	size of data to read
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```

/* read data */

uint32_t temp = 0U;

ospi_parameter_struct ospi_struct = {0};

ospi_read(OSPI_INTERFACE, &ospi_struct, OSPI_MODE, 0x01U, 0x4000U, 0x1000U,
(uint8_t *)&temp, 4);

```

## OSPIWriteRegUsingCSR

The description of OSPIWriteRegUsingCSR is shown as below:

**Table 3-1424. Function OSPIWriteRegUsingCSR**

<b>Function name</b>	OSPIWriteRegUsingCSR
<b>Function prototype</b>	void OSPIWriteRegUsingCSR(uint8_t *WriteBuffer, uint16_t Address, uint8_t Count, uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, interface_mode mode, uint8_t write_cmd, uint8_t read_cmd, addr_inc_mode inc_mode);
<b>Function descriptions</b>	OSPI write register using CSR
<b>Precondition</b>	-
<b>The called functions</b>	ospi_write, ospi_read
<b>Input parameter{in}</b>	
<b>WriteBuffer</b>	pointer to data to be written
<b>Input parameter{in}</b>	
<b>Address</b>	write start address
<b>Input parameter{in}</b>	
<b>Count</b>	size of data to write
<b>Input parameter{in}</b>	
<b>ospi_periph</b>	peripheral OSPIx
<i>OSPIx</i>	x=0,1

Input parameter{in}	
<i>ospi_struct</i>	OSPI parameter initialization struct members of the structure
Input parameter{in}	
<b>interface_mode</b>	SPI interface
<i>SPI_MODE</i>	standard SPI mode
<i>QSPI_MODE</i>	QSPI mode
<i>OSPI_MODE</i>	OSPI mode
Input parameter{in}	
<i>write_cmd</i>	write command
Input parameter{in}	
<i>read_cmd</i>	read command
Input parameter{in}	
<b>inc_mode</b>	address mode
<i>ADDR_NO_INC</i>	static address
<i>ADDR_INC</i>	incrementing address
<i>ADDR_DEC</i>	decrementing address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write data */
```

```
uint8_t *WriteBuffer;
```

```
uint16_t Address = 0x10;
```

```
ospi_parameter_struct ospi_struct = {0};
```

```
OSPIWriteRegUsingCSR(WriteBuffer, Address, 2, OSPI1, &ospi_init_struct, OSPI_MODE,
1U, 0x0BU, 0U);
```

## OSPIReadRegUsingCSR

The description of OSPIReadRegUsingCSR is shown as below:

**Table 3-1425. Function OSPIReadRegUsingCSR**

<b>Function name</b>	OSPIReadRegUsingCSR
<b>Function prototype</b>	void OSPIReadRegUsingCSR (uint8_t *ReadBuffer, uint16_t Address, uint8_t Count, uint32_t ospi_periph, ospi_parameter_struct *ospi_struct, interface_mode mode, uint8_t write_cmd, uint8_t read_cmd, addr_inc_mode inc_mode);
<b>Function descriptions</b>	OSPI read register using CSR
<b>Precondition</b>	-
<b>The called</b>	ospi_write, ospi_read

functions	
Input parameter{in}	
<b>ReadBuffer</b>	pointer to data to be read
Input parameter{in}	
<b>Address</b>	read start address
Input parameter{in}	
<b>Count</b>	size of data to read
Input parameter{in}	
<b>ospi_periph</b>	peripheral OSPIx
<i>OSPIx</i>	x=0,1
Input parameter{in}	
<i>ospi_struct</i>	OSPI parameter initialization struct members of the structure
Input parameter{in}	
<b>interface_mode</b>	SPI interface
<i>SPI_MODE</i>	standard SPI mode
<i>QSPI_MODE</i>	QSPI mode
<i>OSPI_MODE</i>	OSPI mode
Input parameter{in}	
<i>write_cmd</i>	write command
Input parameter{in}	
<i>read_cmd</i>	read command
Input parameter{in}	
<b>inc_mode</b>	address mode
<i>ADDR_NO_INC</i>	static address
<i>ADDR_INC</i>	incrementing address
<i>ADDR_DEC</i>	decrementing address
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* read data */
```

```
uint8_t *ReadBuffer;
```

```
uint16_t Address = 0x10;
```

```
ospi_parameter_struct ospi_struct = {0};
```

```
OSPIReadRegUsingCSR(ReadBuffer, Address, 2, OSPI1, &ospi_init_struct, OSPI_MODE,
1U, 0x0BU, 0U);
```

## OSPIReadRegister

The description of OSPIReadRegister is shown as below:

**Table 3-1426. Function OSPIReadRegister**

<b>Function name</b>	OSPIReadRegister
<b>Function prototype</b>	void OSPIReadRegister(uint8_t *ReadBuffer, uint16_t Address, uint16_t Count);
<b>Function descriptions</b>	OSPI read register
<b>Precondition</b>	-
<b>The called functions</b>	OSPIReadPDRamRegister, OSPIReadRegUsingCSR
<b>Input parameter{in}</b>	
<b>ReadBuffer</b>	pointer to data to be read
<b>Input parameter{in}</b>	
<b>Address</b>	read start address
<b>Input parameter{in}</b>	
<b>Count</b>	size of data to read
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* read data */
uint8_t *ReadBuffer;
uint16_t Address = 0x10;
ospi_parameter_struct ospi_struct = {0};
OSPIReadRegister(ReadBuffer, Address, 2);
```

## OSPIWriteRegister

The description of OSPIWriteRegister is shown as below:

**Table 3-1427. Function OSPIWriteRegister**

<b>Function name</b>	OSPIWriteRegister
<b>Function prototype</b>	void OSPIWriteRegister(uint8_t *WriteBuffer, uint16_t Address, uint16_t Count);
<b>Function descriptions</b>	OSPI write register
<b>Precondition</b>	-
<b>The called functions</b>	OSPIReadPDRamRegister, OSPIReadRegUsingCSR
<b>Input parameter{in}</b>	
<b>WriteBuffer</b>	pointer to data to be written
<b>Input parameter{in}</b>	
<b>Address</b>	write start address

Input parameter{in}	
Count	size of data to write
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* write data */

uint8_t *WriteBuffer;

uint16_t Address = 0x10;

ospi_parameter_struct ospi_struct = {0};

OSPIWriteRegister (WriteBuffer, Address, 2);
```

## 3.41. ESC\_PHY

The phy control by EtherCAT Slave Controller, it contains PHYs A and B, there are identical in functionality. The ESC PHY control registers are listed in chapter [3.41.1](#), the ESC PHY control firmware functions are introduced in chapter [3.41.2](#).

### 3.41.1. Descriptions of Peripheral registers

ESC\_PHY registers are listed in the table shown as below:

**Table 3-1428. ESC\_PHY Registers**

Registers	Descriptions
ESC_MII_CONTROL	ESC MII Management Control register
ESC_MII_STATUS	ESC MII Management Status register
ESC_PHY_ADDR	ESC PHY Address register
ESC_PHY_DATA	ESC PHY Data register
ESC_MII_PDI_STATE	MII Management ECAT Access State register
ESC_PHY_PORT0_STA	PHY Port 0 Status register
ESC_PHY_PORT1_STA	PHY Port 1 Status register

### 3.41.2. Descriptions of Peripheral functions

ESC PHY firmware functions are listed in the table shown as below:

**Table 3-1429. ESC\_PHY firmware function**

Function name	Function description
esc_phy_read	read phy data from specified register
esc_phy_write	write phy data to specified register



Function name	Function description
esc_mmd_read	read phy data from specified MMD register
esc_mmd_write	write phy data to specified MMD register

## esc\_phy\_read

The description of esc\_phy\_read is shown as below:

**Table 3-1430. Function esc\_phy\_read**

Function name	esc_phy_read
Function prototype	unsigned short esc_phy_read(unsigned char phy_addr, unsigned char phy_reg);
Function descriptions	read phy register value
Precondition	-
The called functions	ospi_read
Input parameter{in}	
phy_addr	phy physical address(0-15)
phy_reg	phy register(Address offset)
Output parameter{out}	
-	-
Return value	
unsigned short	res

Example:

```
/* read PHYA control register value */
uint32_t temp_value;
esc_phy_write(0x0,0x1F,0x0);
temp_value = esc_phy_read (0x00,0x00);
```

## esc\_phy\_write

The description of esc\_phy\_write is shown as below:

**Table 3-1431. Function esc\_phy\_write**

Function name	esc_phy_write
Function prototype	void esc_phy_write(unsigned char phy_addr, unsigned char phy_reg, unsigned short phy_data);
Function descriptions	write phy register value
Precondition	-
The called functions	ospi_write
Input parameter{in}	
phy_addr	phy physical address(0-15)
phy_reg	phy register(Address offset)
phy_data	phy data

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

Example:

```
/* write phy PHYA control Register Half Duplex data value */
```

```
esc_phy_write(0x0,0x1F,0x0);
```

```
esc_phy_write(0x0,0x0,0x3000);
```

### esc\_mmd\_read

The description of esc\_mmd\_read is shown as below:

**Table 3-1432. Function esc\_mmd\_read**

Function name	esc_mmd_read
Function prototype	unsigned short esc_mmd_read(unsigned char phy_addr, unsigned char MMD_addr, unsigned short MMD_addr_index);
Function descriptions	read phy data from specified MMD register
Precondition	-
The called functions	esc_phy_write esc_phy_read
Input parameter{in}	
<i>phy_addr</i>	PHY physical address(0-15)
<i>MMD_addr</i>	MMD address(3,7)
<i>MMD_addr_index</i>	MMD address index(Address offset)
Output parameter{out}	
-	-
Return value	
unsigned short	value

Example:

```
/* read phyA specified MMD AN Control Register */
```

```
uint32_t temp_value;
```

```
temp_value = esc_mmd_read(0x00, 0x07,0x00);
```

### esc\_mmd\_write

The description of esc\_mmd\_write is shown as below:

**Table 3-1433. Function esc\_mmd\_write**

<b>Function name</b>	esc_mmd_write
<b>Function prototype</b>	void esc_mmd_write(unsigned char phy_addr, unsigned char MMD_addr, unsigned short MMD_addr_index, unsigned short MMD_data);
<b>Function descriptions</b>	write phy data to specified MMD register
<b>Precondition</b>	-
<b>The called functions</b>	esc_phy_write
<b>Input parameter{in}</b>	
<i>phy_addr</i>	PHY physical address(0-15)
<i>MMD_addr</i>	MMD address(Device Address 3,7)
<i>MMD_addr_index</i>	MMD address index(Address offset)
<i>MMD_data</i>	MMD write data
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write phyA specified MMD AN Control Register Auto-Negotiation Enable */
```

```
esc_mmd_write(0x00,0x07,0x00,0x300);
```

## 3.42. ESC\_PMU

ESC power management unit (PMU) managements device level and module level power saving modes of ESC device. The PMU registers are listed in chapter [3.42.1](#), the PMU firmware functions are introduced in chapter [3.42.2](#).

### 3.42.1. Descriptions of Peripheral registers

ESC\_PMU registers are listed in the table shown as below:

**Table 3-1434. ESC\_PMU Registers**

Registers	Descriptions
PMU_CTRL0	ESC_PMU control register 0
PMU_CTRL1	ESC_PMU control register 1
PMU_PDIREFVAL	ESC_PMU process data interface reference value register

### 3.42.2. Descriptions of Peripheral functions

ESC\_PMU firmware functions are listed in the table shown as below:

**Table 3-1435. ESC\_PMU firmware function**

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

pmu_esc_power_mang_mode_config	configure power management mode
pmu_esc_wake_up_mode_config	configure PMU wake up mode
pmu_esc_led_wm_config	configure PMU LED work mode
pmu_esc_led_inact_stat_config	configure PMU LED inactive state
pmu_esc_ready_stat_get	get device ready
pmu_esc_edwol_stat_get	get energy detect / WoL port status
pmu_esc_edwol_stat_clear	clear energy detect / WoL port status
pmu_esc_fun_config	configure PMU function
pmu_esc_byte_test	test PMU byte
pmu_esc_sleep_mode_enable	enable power management sleep mode
pmu_esc_sleep_mode_disable	disable power management sleep mode

### Enum pmu\_esc\_fun\_enum

**Table 3-1436. Enum pmu\_esc\_fun\_enum**

Member name	Descriptions
PMU_ESC_FUN_EDWOLA	PMU energy detect / WoL port A
PMU_ESC_FUN_EDWOLB	PMU energy detect / WoL port B
PMU_ESC_FUN_ECATCLK	PMU EtherCAT core clock
PMU_ESC_FUN_LEDOUT	PMU LEDs output

### pmu\_esc\_power\_mang\_mode\_config

The description of pmu\_esc\_power\_mang\_mode\_config is shown as below:

**Table 3-1437. Function pmu\_esc\_power\_mang\_mode\_config**

Function name	pmu_esc_power_mang_mode_config
Function prototype	void pmu_esc_power_mang_mode_config(uint32_t pm_mode)
Function descriptions	configure power management mode
Precondition	-
The called functions	-
Input parameter{in}	

<b>pm_mode</b>	power management mode
<i>PMU_ESC_PMMOD0</i>	power management mode 0
<i>PMU_ESC_PMMOD1</i>	power management mode 1
<i>PMU_ESC_PMMOD2</i>	power management mode 2
<i>PMU_ESC_PMMOD3</i>	power management mode 3
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* write PMU ESC register vaule */
```

```
pmu_esc_power_mang_mode_config(PMU_ESC_PMMOD0);
```

### pmu\_esc\_wake\_up\_mode\_config

The description of pmu\_esc\_wake\_up\_mode\_config is shown as below:

**Table 3-1438. Function pmu\_esc\_wake\_up\_mode\_config**

<b>Function name</b>	pmu_esc_wake_up_mode_config
<b>Function prototype</b>	void pmu_esc_wake_up_mode_config(uint32_t wu_mode)
<b>Function descriptions</b>	configure PMU wake up mode
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>wu_mode</b>	wake up mode
<i>PMU_WAKE_UP_MASTER</i>	PMU wake up by master
<i>PMU_WAKE_UP_PME_MASTER</i>	PMU wake up by PME or master
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* configure PMU wake up mode */
```

```
pmu_esc_wake_up_mode_config(PMU_WAKE_UP_MASTER);
```

### pmu\_esc\_led\_wm\_config

The description of pmu\_esc\_led\_wm\_config is shown as below:

Table 3-1439. Function `pmu_esc_led_wm_config`

Function name	<code>pmu_esc_led_wm_config</code>
Function prototype	<code>void pmu_esc_led_wm_config(uint32_t led_wm)</code>
Function descriptions	configure PMU LED work mode
Precondition	-
The called functions	-
Input parameter{in}	
<code>led_wm</code>	the working mode of LEDs
<code>PMU_LED_OPEN_DR AIN</code>	the working mode of LEDs is open-drain / open-source
<code>PMU_LED_PUSH_PUL L</code>	the working mode of LEDs is push-pull
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure PMU LED work mode */
pmu_esc_led_wm_config(PMU_LED_OPEN_DRAIN);
```

### `pmu_esc_led_inact_stat_config`

The description of `pmu_esc_led_inact_stat_config` is shown as below:

Table 3-1440. Function `pmu_esc_led_inact_stat_config`

Function name	<code>pmu_esc_led_inact_stat_config</code>
Function prototype	<code>void pmu_esc_led_inact_stat_config(uint32_t led_inact_stat)</code>
Function descriptions	configure PMU LED inactive state
Precondition	-
The called functions	-
Input parameter{in}	
<code>led_inact_stat</code>	LED inactive state
<code>PMU_LED_INACT_STA T0</code>	0 is inactive state
<code>PMU_LED_INACT_STA T1</code>	1 is inactive state
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure PMU LED inactive state */
```

```
pmu_esc_led_inact_stat_config(PMU_LED_INACT_STAT0);
```

### pmu\_esc\_ready\_stat\_get

The description of pmu\_esc\_ready\_stat\_get is shown as below:

**Table 3-1441. Function pmu\_esc\_ready\_stat\_get**

<b>Function name</b>	pmu_esc_ready_stat_get
<b>Function prototype</b>	FlagStatus pmu_esc_ready_stat_get(void)
<b>Function descriptions</b>	get device ready status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* get device ready status */
```

```
FlagStatus state = pmu_esc_ready_stat_get();
```

### pmu\_esc\_edwol\_stat\_get

The description of pmu\_esc\_edwol\_stat\_get is shown as below:

**Table 3-1442. Function pmu\_esc\_edwol\_stat\_get**

<b>Function name</b>	pmu_esc_edwol_stat_get
<b>Function prototype</b>	FlagStatus pmu_esc_edwol_stat_get(uint32_t edwol_port)
<b>Function descriptions</b>	get energy detect / WoL port status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>edwol_port</b>	energy detect / WoL port
<i>PMU_EDWOL_PORTA</i>	energy detect / WoL portA
<i>PMU_EDWOL_PORTB</i>	energy detect / WoL portB
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* get energy detect / WoL port status */
```

```
FlagStatus state = pmu_esc_edwol_stat_get();
```

### pmu\_esc\_edwol\_stat\_clear

The description of pmu\_esc\_edwol\_stat\_clear is shown as below:

**Table 3-1443. Function pmu\_esc\_edwol\_stat\_clear**

<b>Function name</b>	pmu_esc_edwol_stat_clear
<b>Function prototype</b>	void pmu_esc_edwol_stat_clear(uint32_t edwol_port)
<b>Function descriptions</b>	clear energy detect / WoL port status
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>edwol_port</b>	energy detect / WoL port
<i>PMU_EDWOL_PORTA</i>	energy detect / WoL portA
<i>PMU_EDWOL_PORTB</i>	energy detect / WoL portB
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
<b>FlagStatus</b>	SET or RESET

Example:

```
/* clear energy detect / WoL port status */
pmu_esc_edwol_stat_clear(PMU_EDWOL_PORTA);
```

### pmu\_esc\_fun\_config

The description of pmu\_esc\_fun\_config is shown as below:

**Table 3-1444. Function pmu\_esc\_fun\_config**

<b>Function name</b>	pmu_esc_fun_config
<b>Function prototype</b>	void pmu_esc_fun_config(pmu_esc_fun_enum fun_type, ControlStatus newvalue)
<b>Function descriptions</b>	configure PMU function
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>fun_type</b>	PMU fuction, refer to <a href="#">Table 3-1436. Enum pmu_esc_fun_enum</a>
<b>Input parameter{in}</b>	
<b>newvalue</b>	ENABLE or DISABLE
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-



Example:

```
/* configure PMU function */
pmu_esc_fun_config(PMU_ESC_FUN_EDWOLB, ENABLE);
```

### pmu\_esc\_byte\_test

The description of pmu\_esc\_byte\_test is shown as below:

**Table 3-1445. Function pmu\_esc\_byte\_test**

<b>Function name</b>	pmu_esc_byte_test
<b>Function prototype</b>	FlagStatus pmu_esc_byte_test(void)
<b>Function descriptions</b>	test PMU byte
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Input parameter{in}</b>	
newvalue	ENABLE or DISABLE
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* test PMU byte */
pmu_esc_byte_test();
```

### pmu\_esc\_sleep\_mode\_enable

The description of pmu\_esc\_sleep\_mode\_enable is shown as below:

**Table 3-1446. Function pmu\_esc\_sleep\_mode\_enable**

<b>Function name</b>	pmu_esc_sleep_mode_enable
<b>Function prototype</b>	void pmu_esc_sleep_mode_enable(void)
<b>Function descriptions</b>	enable power management 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 power management sleep mode */
pmu_esc_sleep_mode_enable();
```

### pmu\_esc\_sleep\_mode\_disable

The description of pmu\_esc\_sleep\_mode\_disable is shown as below:

**Table 3-1447. Function pmu\_esc\_sleep\_mode\_disable**

<b>Function name</b>	pmu_esc_sleep_mode_disable
<b>Function prototype</b>	void pmu_esc_sleep_mode_disable(void)
<b>Function descriptions</b>	disable power management 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 power management sleep mode */
pmu_esc_sleep_mode_disable();
```

## 3.43. ESC\_SYSCFG

ESC system configuration (ESC\_SYSCFG) is used for configuring the ESC module ID and the ESC kernel controller. Section [3.43.1](#) describes the register list of ESC\_SYSCFG, and Section [3.43.2](#) explains the WWDGT library functions.

### 3.43.1. Descriptions of Peripheral registers

ESC\_SYSCFG registers are listed in the table shown as below:

**Table 3-1448. ESC\_SYSCFG Registers**

Registers	Descriptions
ESC_EF_CHIP_ID	ESC_EF_CHIP_ID register
ESC_EFUSE_UID_READ	ESC_EFUSE_UID_REG register
ESC_CCTL_DATA	ESC_CCTL_DATA register
ESC_CCTL_CMD	ESC_CCTL_CMD register
ESC_PRAM_FIFO_	ESC_PRAM_FIFO_DR register

Registers	Descriptions
DR	
ESC_PRAM_ALR	ESC_PRAM_ALR_OFFSET register
ESC_PRAM_CR	ESC_PRAM_CR_OFFSET register
ESC_PRAM_FIFO_DW	ESC_PRAM_FIFO_DW register
ESC_PRAM_ALW	ESC_PRAM_ALW register
ESC_PRAM_CW	ESC_PRAM_CW register
ESC_OPB_CS	ESC_OPB_CS register
ESC_SYSCFG_CFG0	ESC_SYSCFG_CFG0 register
ESC_SYSCFG_CHIPID	ESC_SYSCFG_CHIPID register
ESC_SYSCFG_CHIPVER	ESC_SYSCFG_CHIPVER register
ESC_SYSCFG_RESERVED	ESC_SYSCFG_RESERVED register

### 3.43.2. Descriptions of Peripheral functions

ESC\_SYSCFG firmware functions are listed in the table shown as below:

**Table 3-1449. ESC\_SYSCFG firmware function**

Function name	Function description
syscfg_get_chip_id	read chip id
syscfg_get_chip_version	read chip version
syscfg_efuse_read	read efuse registers

#### syscfg\_get\_chip\_id

The description of syscfg\_get\_chip\_id is shown as below:

**Table 3-1450. Function syscfg\_get\_chip\_id**

Function name	syscfg_get_chip_id
Function prototype	uint32_t syscfg_get_chip_id(void);
Function descriptions	read chip id
Precondition	-
The called functions	-
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
uin32_t	temp_value

Example:

```
uint32_t chip_id = 0;

/* get chip id */

chip_id = syscfg_get_chip_id();
```

### syscfg\_get\_chip\_version

The description of syscfg\_get\_chip\_version is shown as below:

**Table 3-1451. Function syscfg\_get\_chip\_version**

<b>Function name</b>	syscfg_get_chip_version
<b>Function prototype</b>	uint32_t syscfg_get_chip_version(void);
<b>Function descriptions</b>	get chip version
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	temp_value

Example:

```
uint32_t chip_ver = 0;

/* get chip version */

chip_ver = syscfg_get_chip_version();
```

### syscfg\_efuse\_read

The description of syscfg\_efuse\_read is shown as below:

**Table 3-1452. Function syscfg\_efuse\_read**

<b>Function name</b>	syscfg_efuse_read
<b>Function prototype</b>	void syscfg_efuse_read(uint16_t counter_value);
<b>Function descriptions</b>	get the content of EFUSE register
<b>Precondition</b>	-
<b>The called functions</b>	-
<b>Input parameter{in}</b>	
<b>reg</b>	<b>accessed register</b>
ESC_EF_CHIP_ID	get ESC_EF_CHIP_ID
ESC_EFUSE_UID_READ	get ESC_EFUSE_UID_READ
<b>Output parameter{out}</b>	

-	-
Return value	
uint32_t	temp_value

Example:

```
uint32_t temp_value = 0;
```

```
/* read efuse content */
```

```
temp_value = syscfg_efuse_read(ESC_EF_CHIP_ID);
```

## 3.44. ESC\_TIMER

The timers have a counter that can be used as an unsigned counter and supports both input capture and output compare. ESC\_TIMER is divided into two sorts: basic timer (ESC\_BASIC\_TIMER), free-running counter (ESC\_FRC). The specific functions of different types of timer are different. The ESC\_TIMER registers are listed in chapter [3.44.1](#), the ESC\_TIMER firmware functions are introduced in chapter [3.44.2](#).

### 3.44.1. Descriptions of Peripheral registers

ESC\_TIMER registers are listed in the table shown as below:

**Table 3-1453. ESC\_TIMER Registers**

Registers	Descriptions
ESC_TIMER_CTL0	ESC BASIC TIMER control register 0
ESC_TIMER_CNT	ESC BASIC TIMER counter register
ESC_FRC_CNT	ESC FRC counter register

### 3.44.2. Descriptions of Peripheral functions

ESC\_TIMER firmware functions are listed in the table shown as below:

**Table 3-1454. ESC\_TIMER firmware function**

Function name	Function description
esc_timer_enable	enable a ESC_BASIC_TIMER
esc_timer_disable	disable a ESC_BASIC_TIMER
esc_timer_autoreload_value_config	configure ESC BASIC TIMER autoreload register value
esc_timer_counter_read	read ESC BASIC TIMER counter value
esc_frc_counter_read	read ESC FRC counter value

#### esc\_timer\_enable

The description of esc\_timer\_enable is shown as below:

**Table 3-1455. Function esc\_timer\_enable**

<b>Function name</b>	esc_timer_enable
<b>Function prototype</b>	void esc_timer_enable(void);
<b>Function descriptions</b>	enable a ESC_BASIC_TIMER
<b>Precondition</b>	-
<b>The called functions</b>	ospi_read ospi_write
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
-	-

Example:

```
/* enable a ESC_BASIC_TIMER */
esc_timer_enable();
```

### esc\_timer\_disable

The description of esc\_timer\_disable is shown as below:

**Table 3-1456. Function esc\_timer\_disable**

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

Example:

```
/* disable a ESC_BASIC_TIMER */
esc_timer_disable();
```

### esc\_timer\_autoreload\_value\_config

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

Table 3-1457. Function `esc_timer_autoreload_value_config`

Function name	<code>esc_timer_autoreload_value_config</code>
Function prototype	<code>void esc_timer_autoreload_value_config(uint16_t autoreload);</code>
Function descriptions	configure ESC BASIC TIMER autoreload register value
Precondition	-
The called functions	<code>ospi_write</code>
Input parameter{in}	
<b>autoreload</b>	the counter auto-reload value
<i>uint16_t</i>	0x0000 – 0xFFFF
Output parameter{out}	
-	-
Return value	
-	-

Example:

```
/* configure ESC_BASIC_TIMER autoreload register value */
esc_timer_autoreload_value_config(100);
```

### `esc_timer_counter_read`

The description of `esc_timer_counter_read` is shown as below:

Table 3-1458. Function `esc_timer_counter_read`

Function name	<code>esc_timer_counter_read</code>
Function prototype	<code>uint32_t esc_timer_counter_read(void);</code>
Function descriptions	read ESC BASIC TIMER counter value
Precondition	-
The called functions	<code>ospi_read</code>
Input parameter{in}	
-	-
Output parameter{out}	
-	-
Return value	
<b>uint32_t</b>	counter value 0~0xFFFF

Example:

```
/* read ESC_BASIC_TIMER counter value */
uint32_t val = 0;
val = esc_timer_counter_read();
```

## esc\_frc\_counter\_read

The description of esc\_frc\_counter\_read is shown as below:

**Table 3-1459. Function esc\_frc\_counter\_read**

<b>Function name</b>	esc_frc_counter_read
<b>Function prototype</b>	uint32_t esc_frc_counter_read(void);
<b>Function descriptions</b>	read ESC FRC counter value
<b>Precondition</b>	-
<b>The called functions</b>	ospi_read
<b>Input parameter{in}</b>	
-	-
<b>Output parameter{out}</b>	
-	-
<b>Return value</b>	
uint32_t	counter value 0~0xFFFFFFFF

Example:

```
/* read ESC_FRC counter value */

uint32_t val = 0;

val = esc_frc_counter_read();
```



## 4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Nov.8, 2024
1.1	<ol style="list-style-type: none"> <li>1. Delete PMU_SMPS_1V8_SUPPLIES_LDO, PMU_SMPS_2V5_SUPPLIES_LDO, PMU_SMPS_1V8_SUPPLIES_EXT_AND_LDO, PMU_SMPS_2V5_SUPPLIES_EXT_AND_LDO, PMU_SMPS_1V8_SUPPLIES_EXT and PMU_SMPS_2V5_SUPPLIES_EXT in <b><u>Table 3-864. Function</u></b> <b><u>pmu smps ldo supply config.</u></b></li> </ol>	Jan.24, 2025
1.2	<ol style="list-style-type: none"> <li>1. Modify the description in <b><u>chapter 3.2.</u></b></li> <li>2. Delete ob_tcm_shared_ram_size_get, add ob_itcm_shared_ram_size_get and ob_dtcn_shared_ram_size_get in <b><u>chapter 3.16.</u></b></li> <li>3. Delete i2c_nack_disable in <b><u>chapter 3.20.</u></b></li> </ol>	Aug.7, 2025

## 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.